

Puppet : Solution de gestion de fichier de configuration

Sommaire

- 1 Introduction
- 2 Hiérarchie de Puppet
- 3 Installation
 - 3.1 Serveur
 - 3.1.1 Fonctionnement de puppet avec Mongrel et NGINX
 - 3.1.1.1 Installation
 - 3.1.1.2 Configuration
 - 3.2 Clients
 - 3.2.1 Debian
 - 3.2.2 Red Hat
 - 3.2.3 Solaris
- 4 Configuration
 - 4.1 Serveur
 - 4.1.1 auth.conf
 - 4.1.2 autosign.conf
 - 4.1.3 fileserv.conf
 - 4.1.4 manifests
 - 4.1.4.1 common.pp
 - 4.1.4.2 modules.pp
 - 4.1.4.3 site.pp
 - 4.1.5 puppet.conf
 - 4.2 Client
 - 4.2.1 puppet.conf
 - 4.2.1.1 Debian / Red Hat
 - 4.2.1.2 Solaris
- 5 Le langage
 - 5.1 Les fonctions
 - 5.2 Installer des packages
 - 5.3 Inclure / Exclure des modules
 - 5.4 Les templates
 - 5.4.1 Les inline-templates
 - 5.5 Les Facters
 - 5.6 Informations dynamiques
 - 5.7 Les Parsers
 - 5.8 Du Ruby dans vos manifests
 - 5.9 Ajouter une variable Ruby dans les manifests
 - 5.10 Les classes
 - 5.11 Utilisation des tables de hash
 - 5.12 Les regex
 - 5.13 Substitution
 - 5.14 Notify et Require
 - 5.15 L'opérateur +>
 - 5.16 Vérifier le numéro de version d'un soft
 - 5.17 Les ressources virtuelles
 - 5.18 Suppression d'un fichier évoluée
- 6 Les modules
 - 6.1 Initialisation d'un module
 - 6.2 Le module initiale (base)
 - 6.2.1 init.pp
 - 6.2.2 vars.pp
 - 6.2.3 functions.pp
 - 6.2.4 roles.pp
 - 6.2.5 servers.pp
 - 6.2.6 Parser
- 7 Exemples de modules
 - 7.1 Puppet
 - 7.1.1 init.pp
 - 7.1.2 redhat.pp
 - 7.1.3 files
 - 7.2 resolvconf
 - 7.2.1 init.pp
 - 7.2.2 redhat.pp
 - 7.2.3 templates
 - 7.3 packages_defaults
 - 7.3.1 init.pp
 - 7.3.2 redhat.pp
 - 7.4 configurations_defaults
 - 7.4.1 init.pp
 - 7.4.2 common.pp
 - 7.4.3 redhat.pp
 - 7.4.4 security.pp
 - 7.4.5 facter
 - 7.4.5.1 passwd_algorithm.rb
 - 7.4.5.2 kernel_rights.rb
 - 7.4.5.3 get_network_infos.rb
 - 7.4.6 network.pp
 - 7.4.7 templates
 - 7.5 OpenSSH - 1
 - 7.5.1 init.pp
 - 7.5.2 redhat.pp
 - 7.5.3 common.pp
 - 7.5.4 facter
 - 7.5.5 ssh_keys.pp

- 7.5.6 files
 - 7.6 OpenSSH - 2
 - 7.6.1 init.pp
 - 7.6.2 templates
 - 7.7 SELinux
 - 7.7.1 init.pp
 - 7.7.2 redhat.pp
 - 7.8 Grub
 - 7.8.1 init.pp
 - 7.8.2 redhat.pp
 - 7.9 Kdump
 - 7.9.1 init.pp
 - 7.9.2 redhat.pp
 - 7.10 Tools
 - 7.10.1 init.pp
 - 7.10.2 redhat.pp
 - 7.10.3 files
 - 7.11 Timezone
 - 7.11.1 init.pp
 - 7.11.2 redhat.pp
 - 7.11.3 templates
 - 7.12 NTP
 - 7.12.1 init.pp
 - 7.12.2 redhat.pp
 - 7.12.3 files
 - 7.13 MySecureShell
 - 7.13.1 init.pp
 - 7.13.2 redhat.pp
 - 7.13.3 files
 - 7.14 OpenLDAP client & Server
 - 7.14.1 init.pp
 - 7.14.2 redhat.pp
 - 7.14.3 facter
 - 7.14.3.1 generated_ldap_servers.rb
 - 7.14.3.2 current_ldap_servers.rb
 - 7.14.4 Parser
 - 7.14.4.1 dns2ip.rb
 - 7.14.5 redhat_ldapclient.pp
 - 7.14.6 redhat_ldapservers.pp
 - 7.14.7 files
 - 7.15 sudo
 - 7.15.1 init.pp
 - 7.15.1.1 Amélioration d'un module
 - 7.15.2 files
 - 7.16 snmpd
 - 7.16.1 init.pp
 - 7.16.2 redhat.pp
 - 7.16.3 files
 - 7.17 Postfix
 - 7.17.1 init.pp
 - 7.17.2 redhat.pp
 - 7.17.3 templates
 - 7.17.4 files
 - 7.18 Nsswitch
 - 7.18.1 facter
 - 7.18.2 init.pp
 - 7.18.3 templates
 - 7.19 Nagios NRPE + plugins
 - 7.19.1 init.pp
 - 7.19.2 files
 - 7.19.3 templates
 - 7.20 Munin
 - 7.20.1 Munin Interfaces
 - 7.21 Mcollective
 - 7.21.1 init.pp
 - 7.21.2 common.pp
 - 7.21.3 redhat.pp
 - 7.21.4 files
 - 7.21.4.1 RedHat.server.cfg
 - 7.22 Bind
 - 7.22.1 init.pp
 - 7.22.2 redhat.pp
 - 7.22.3 files
 - 7.22.4 templates
 - 7.23 Importation d'un module
 - 7.23.1 Importer tous les modules d'un coup
- 8 Utilisation
 - 8.1 Certificats
 - 8.1.1 Création d'un certificat
 - 8.1.2 Ajout d'un client puppet au serveur
 - 8.1.3 Synchroniser un client
 - 8.1.3.1 Simuler
 - 8.1.4 Révoquer un certificat
 - 8.1.5 Révoquer tous les certificats du serveur
 - 8.2 Surveillance des processus
 - 8.2.1 Détermination de l'état des noeuds
- 9 Utilisation avancée
 - 9.1 Vérifier la syntaxe de ses .pp
 - 9.2 Outrepasser des restrictions
 - 9.3 Désactiver une ressource
 - 9.4 Pre et Post puppet run

- 9.5 CFT
- 9.6 Générer un manifest depuis un système existant
- 9.7 Puppet Push
- 9.8 MCollective
- 10 FAQ
 - 10.1 err: Could not retrieve catalog from remote server: hostname was not match with the server certificate
- 11 Ressources

1 Introduction

Puppet est une application très pratique... C'est ce que l'on pourrait retrouver dans les entreprises avec de grands volumes de serveurs, où le système d'information est « industrialisé ».

Puppet permet d'automatiser un grand nombre de tâche d'administration, comme l'installation de logiciels, de services ou encore de modifier des fichiers.

Puppet permet de faire cela de manière centralisée ce qui permet d'administrer et de mieux contrôler un grand nombre de serveur hétérogènes ou homogènes.

Puppet fonctionne en mode Client / Serveur.

Sur chaque machine un client va être installé et c'est lui qui va contacter le PuppetMaster, le serveur, par le biais de communication HTTPS, et donc SSL, un système pki est fourni.

Puppet a été développé en Ruby le rendant multiplateforme : bsd (free, macos ...) linux (redhat, debian, suse ...) sun (opensolaris ...)

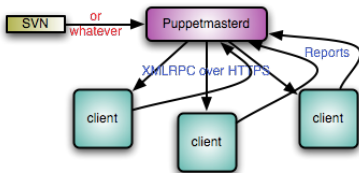
Reductive Labs (<http://reductivelabs.com/>), la société éditant Puppet, a développé un produit complémentaire, nommé Factor.

Cette application permet de lister des éléments propres aux systèmes administrés, comme le nom de machine, l'adresse ip, la distribution, des variables d'environnement utilisables dans les templates de puppet.

Puppet gérant des templates, on peut rapidement comprendre l'utilité de Factor, si par exemple on gère une ferme de serveurs de messagerie, qui nécessite un paramétrage contenant par exemple le nom de la machine. Là un template combiné avec des variables d'environnements s'avère tout à fait utile.

Enfin bref Puppet combiné à Factor me semble une solution très intéressante pour simplifier l'administration de systèmes.

Voici un schéma de fonctionnement de puppet :



Pour la configuration de Puppet, si vous souhaitez utiliser un IDE, il existe Geppetto (<http://www.puppetlabs.com/blog/geppetto-a-puppet-ide/>) . Je vous le recommande d'ailleurs, ça vous évitera bien des soucis de syntaxe.

Les documentations pour des version antérieures à ce celle ci sont disponible ici :

- Puppet 0.25.4

2 Hiérarchie de Puppet

Avant d'aller plus loin, j'ai pompé depuis le site officiel le fonctionnement de l'arborescence de puppet :

All Puppet data files (modules, manifests, distributable files, etc) should be maintained in a Subversion or CVS repository (or your favorite Version Control System). The following hierarchy describes the layout one should use to arrange the files in a maintainable fashion:

- `/manifests/`: this directory contains files and subdirectories that determine the manifest of individual systems but do not logically belong to any particular module. Generally, this directory is fairly thin and alternatives such as the use of LDAP or other external node tools can make the directory even thinner. This directory contains the following special files:
 - `site.pp`: first file that the Puppet Master parses when determining a server's catalog. It imports all the underlying subdirectories and the other special files in this directory. It also defines any global defaults, such as package managers. See `sample site.pp`.
 - `templates.pp`: defines all template classes. See also `terminology:template classes`. See `sample templates.pp`.
 - `nodes.pp`: defines all the nodes if not using an external node tool. See `sample nodes.pp`.
- `/modules/{modulename}/`: houses puppet modules in subdirectories with names matching that of the module name. This area defines the general building blocks of a server and contains modules such as `for openssh`, which will generally define classes `openssh::client` and `openssh::server` to setup the client and server respectively. The individual module directories contains subdirectories for manifests, distributable files, and templates. See `modules organization`, `terminology:module`.
- `/modules/user/`: A special module that contains manifests for users. This module contains a special subclass called `user::virtual` which declares all the users that might be on a given system in a virtual way. The other subclasses in the user module are classes for logical groupings, such as `user::unixadmins`, which will realize the individual users to be included in that group. See also `naming conventions`, `terminology:realize`.
- `/services/`: this is an additional modules area that is specified in the module path for the puppetmaster. However, instead of generic modules for individual services and bits of a server, this module area is used to model servers specific to enterprise level infrastructure services (core infrastructure services that your IT department provides, such as `www`, `enterprise directory`, `file server`, etc). Generally, these classes will include the modules out of `/modules/` needed as part of the catalog (such as `openssh::server`, `postfix`, `user::unixadmins`, etc). The files section for these modules is used to distribute configuration files specific to the enterprise infrastructure service such as `openldap schema files` if the module were for the enterprise directory. To avoid namespace collision with the general modules, it is recommended that these modules/classes are prefixed with `s_` (e.g. `s_ldap` for the enterprise directory server module)
- `/clients/`: similar to the `/services/` module area, this area is used for modules related to modeling servers for external clients (departments outside your IT department). To avoid namespace collision, it is recommended that these modules/classes are prefixed with `c_`.
- `/notes/`: this directory contains notes for reference by local administrators.
- `/plugins/`: contains custom types programmed in Ruby. See also `terminology:plugin-type`.
- `/tools/`: contains scripts useful to the maintenance of Puppet.

3 Installation

3.1 Serveur

La version utilisée du master doit être la même que celle des postes clients. Il est très fortement recommandé d'utiliser une version supérieure ou égale à 0.25.4 (correctif de nombreux problèmes de performance). Pour cela, sur Debian, il faudra installer la version disponible en `squeeze/lenny-backport` ou supérieur, et la bloquer pour éviter qu'une upgrade malencontreuse ne change sa version (utilisation des "pin locks"). Nous allons opter ici pour la version donnée sur le site officiel de Puppet et donc utiliser la version 2.7.9.

Puppet

Software version	2.7.12+
Operating System	Debian 6
Website	Puppet Website (http://puppetlabs.com/) Clients OS: Debian 6 Solaris 10 RHEL 6
Others	

Pour le moment, il faut configurer le fichier /etc/hosts avec l'ip du serveur :

```
192.168.0.93 puppet-prd.deimos.fr puppet
```

Note : Vérifiez que l'horloge du puppetmaster (et les clients aussi bien sûr) est bien à jour/synchronisée. Il peut y avoir un problème avec les certificats qui seront non reconnus/acceptés si il y a un décalage (faire un `dpkg-reconfigure tz-data`).

Configurez le repository officiel de puppet si vous souhaitez la dernière version, sinon sautez cette étape pour installer la version qui est fournie par votre distribution :

```
0. echo "# Puppet official repository"
1. deb http://apt.puppetlabs.com/ wheezy main
2. " >> /etc/apt/sources.list
```

Importez la clé GPG du repository :

```
0. gpg --keyserver pgpkeys.mit.edu --recv-key 1054B7A24BD6EC30
1. gpg --a --export 1054B7A24BD6EC30 | apt-key add -
```

Et ensuite, nous mettez à jour :

```
0. aptitude update
```

Puis installer puppetmaster :

```
aptitude install puppetmaster mongrel
```

On peut vérifier que puppetmaster est bien installé en lançant 'facter' (voir si elle retourne bien quelque chose) ou la présence des fichiers SSL (dans /var/lib/puppet).

3.1.1 Fonctionnement de puppet avec Mongrel et NGINX

3.1.1.1 Installation

Il faut installer nginx :

```
aptitude install nginx
```

3.1.1.2 Configuration

Modification du fichier /etc/default/puppetmaster :

```
# Defaults for puppetmaster - sourced by /etc/init.d/puppet
# Start puppet on boot?
START=yes
# Startup options
DAEMON_OPTS=""
# What server type to run
# Options:
#   webrick (default, cannot handle more than ~30 nodes)
#   mongrel (scales better than webrick because you can run
#           multiple processes if you are getting
#           connection-reset or End-of-file errors, switch to
#           mongrel. Requires front-end web-proxy such as
#           apache, nginx, or pound)
# See: http://redactive.labs.com/trac/puppet/wiki/UsingMongrel
SERVERTYPE=mongrel
# How many puppetmaster instances to start? Its pointless to set this
# higher than 1 if you are not using mongrel.
PUPPETMASTERS=4
# What port should the puppetmaster listen on (default: 8140). If
# PUPPETMASTERS is set to a number greater than 1, then the port for
# the first puppetmaster will be set to the port listed below, and
```

```

# further instances will be incremented by one
#
# NOTE: if you are using mongrel, then you will need to have a
# front-end web-proxy (such as apache, nginx, pound) that takes
# incoming requests on the port your clients are connecting to
# (default is: 8140), and then passes them off to the mongrel
# processes. In this case it is recommended to run your web-proxy on
# port 8140 and change the below number to something else, such as
# 18140.
PORT=18140

```

Après (re-)lancement du démon, on doit pouvoir voir les sockets attachées :

```

netstat

0. > netstat -pvltpn
1. Connexions Internet actives (seulement serveurs)
2. Proto Recv-Q Send-Q Adresse locale Adresse distante Etat PID/Program name
3. tcp 0 0 0 0.0.0.0:41736 0.0.0.0:* LISTEN 2029/rpc.statd
4. tcp 0 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 2018/portmap
5. tcp 0 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 2333/sshd
6. tcp 0 0 0 127.0.0.1:18140 0.0.0.0:* LISTEN 10059/ruby
7. tcp 0 0 0 127.0.0.1:18141 0.0.0.0:* LISTEN 10082/ruby
8. tcp 0 0 0 127.0.0.1:18142 0.0.0.0:* LISTEN 10104/ruby
9. tcp 0 0 0 127.0.0.1:18143 0.0.0.0:* LISTEN 10126/ruby
10. tcp6 0 0 0 :::22 :::* LISTEN 2333/sshd

```

On ajoute les lignes suivante dans le fichier /etc/puppet/puppet.conf :

```

/etc/puppet/puppet.conf

0. [main]
1. logdir=/var/log/puppet
2. vardir=/var/lib/puppet
3. ssldir=/var/lib/puppet/ssl
4. rundir=/var/run/puppet
5. factpath=$vardir/lib/facter
6. templatedir=$confdir/templates
7.
8. [master]
9. # These are needed when the puppetmaster is run by passenger
10. # and can safely be removed if webrick is used.
11. ssl_client_header = HTTP_X_SSL_SUBJECT
12. ssl_client_verify_header = SSL_CLIENT_VERIFY
13. report = true
14.
15. [main]
16. pluginsync = true

```

On modifie la configuration suivante dans /etc/nginx.conf :

```

/etc/nginx/nginx.conf

0. user www-data;
1. worker_processes 4;
2.
3. error_log /var/log/nginx/error.log;
4. pid /var/run/nginx.pid;
5.
6. events {
7.     worker_connections 1024;
8.     # multi_accept on;
9. }
10.
11. http {
12.     #include /etc/nginx/mime.types;
13.     default_type application/octet-stream;
14.
15.     access_log /var/log/nginx/access.log;
16.
17.     sendfile on;
18.     tcp_nopush on;
19.
20.     # Look at TLB size in /proc/cpuinfo (Linux) for the 4k pagesize
21.     large_client_header_buffers 16 4k;
22.     proxy_buffers 128 4k;
23.
24.     #keepalive_timeout 0;
25.     keepalive_timeout 65;
26.     tcp_nodelay on;
27.
28.     gzip on;
29.     gzip_disable "MSIE [1-6]\.(?!.*SV1)*";
30.
31.     include /etc/nginx/conf.d/*.conf;
32.     include /etc/nginx/sites-enabled/*;
33. }

```

Et on ajoute cette configuration pour puppet :

```

/etc/nginx/sites-available/puppetmaster.conf

0. upstream puppet-prd.deimos.fr {
1.     server 127.0.0.1:18140;
2.     server 127.0.0.1:18141;
3.     server 127.0.0.1:18142;
4.     server 127.0.0.1:18143;
5. }
6.
7. server {
8.     listen 8140;
9.
10.     ssl on;

```

```

11. ssl_certificate      /var/lib/puppet/ssl/certs/puppet-prd.pem;
12. ssl_certificate_key /var/lib/puppet/ssl/private_keys/puppet-prd.pem;
13. ssl_client_certificate /var/lib/puppet/ssl/ca/ca.crt.pem;
14. ssl_ciphers         SSLv2:-LOW:-EXPORT:RC4+RSA;
15. ssl_session_cache  shared:SSL:8m;
16. ssl_session_timeout 5m;
17.
18. ssl_verify_client  optional;
19.
20. # obey to the Puppet CRL
21. ssl_crl             /var/lib/puppet/ssl/ca/ca.crl.pem;
22.
23. root                /var/empty;
24. access_log          /var/log/nginx/access-8140.log;
25. #rewrite_log        /var/log/nginx/rewrite-8140.log;
26.
27. # Variables
28. # $ssl_cipher returns the line of those utilized it is cipher for established SSL-connection
29. # $ssl_client_serial returns the series number of client certificate for established SSL-connection
30. # $ssl_client_s_dn returns line subject DN of client certificate for established SSL-connection
31. # $ssl_client_i_dn returns line issuer DN of client certificate for established SSL-connection
32. # $ssl_protocol returns the protocol of established SSL-connection
33.
34. location / {
35.     proxy_pass      http://puppet-prd.demos.fr;
36.     proxy_redirect  off;
37.     proxy_set_header Host                $host;
38.     proxy_set_header X-Real-IP          $remote_addr;
39.     proxy_set_header X-Forwarded-For    $proxy_add_x_forwarded_for;
40.     proxy_set_header X-Client-DN        $ssl_client_s_dn;
41.     proxy_set_header X-Client-Verify    $ssl_client_verify;
42.     proxy_set_header X-Subject           $ssl_client_s_dn;
43.     proxy_set_header X-SSL-Issuer        $ssl_client_i_dn;
44.     proxy_read_timeout 65;
45. }
46. }
47. }

```

Ensuite on crée le lien symbolique pour appliquer la configuration :

```

ln
cd /etc/nginx/sites-available
ln -s /etc/nginx/sites-enabled/puppetmaster .

```

Et ensuite on redémarre le serveur Nginx.

Pour vérifier que les daemons tournent correctement, vous devez avoir les sockets suivantes ouvertes :

```

netstat
0. > netstat -vlptn
1. Connexions Internet actives (seulement serveurs)
2. Proto Recv-Q Send-Q Adresse locale Adresse distante Etat PID/Program name
3. tcp 0 0 0.0.0.0:41736 0.0.0.0:* LISTEN 2029/rpc.statd
4. tcp 0 0 0.0.0.0:8140 0.0.0.0:* LISTEN 10293/nginx
5. tcp 0 0 0.0.0.0:8141 0.0.0.0:* LISTEN 10293/nginx
6. tcp 0 0 0.0.0.0:1111 0.0.0.0:* LISTEN 2018/portmap
7. tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 2333/sshd
8. tcp 0 0 127.0.0.1:18140 0.0.0.0:* LISTEN 10059/ruby
9. tcp 0 0 127.0.0.1:18141 0.0.0.0:* LISTEN 10082/ruby
10. tcp 0 0 127.0.0.1:18142 0.0.0.0:* LISTEN 10104/ruby
11. tcp 0 0 127.0.0.1:18143 0.0.0.0:* LISTEN 10126/ruby
12. tcp6 0 0 :::22 :::* LISTEN 2333/sshd

```

3.2 Clients

Pour les clients, c'est simple aussi. Mais avant ajoutez la ligne du serveur dans le fichier hosts :

```

/etc/hosts
...
192.168.0.93 puppet-prd.demos.fr puppet

```

Ceci n'est pas obligatoire si vos noms DNS sont correctement configurés.

3.2.1 Debian

Si vous souhaitez utiliser la dernière version :

```

0. echo "# Puppet official repository
1. deb http://apt.puppetlabs.com/ wheezy main
2. " >> /etc/apt/sources.list

```

Importez la clé GPG du repository :

```

gpg

```

```
0. gpg --keyserver pgpkeys.mit.edu --recv-key 1054B7A24BD6EC30
1. gpg --a --export 1054B7A24BD6EC30 | apt-key add -
```

Et ensuite, nous mettez à jour :

```
aptitude
```

```
0. aptitude update
```

Vérifier que le fichier /etc/hosts contient bien le hostname de la machine cliente, puis installez puppet :

```
apt-get
```

```
apt-get install puppet
```

3.2.2 Red Hat

Tout comme Debian, il existe un repo yum sur Red Hat :

```
/etc/yum.repos.d/puppet.repo
```

```
0. [puppet-repo]
1. name=Puppet Repository
2. baseurl=http://yum.puppetlabs.com/el/6/products/x86_64/
3. enabled=1
4. gpgcheck=0
```

Puis on installe :

```
yum
```

```
0. yum install puppet
```

3.2.3 Solaris

Le client Puppet dans la version stable de blastwave est trop ancien (0.23). Il faudra donc installer Puppet (et Factor) par l'intermédiaire du gestionnaire standard de Ruby: gem. Pour cela, il faudra au préalable installer ruby avec la commande suivante:

```
pkg-get
```

```
pkg-get -i ruby
```

WARNING

Vérifier que rubygems n'est pas déjà installé, sinon le supprimer :

```
pkg-get
```

```
pkg-get -r rubygems
```

puis installer une version plus à jour à partir des sources:

```
wget http://rubyforge.org/frs/download.php/45905/rubygems-1.3.1.tgz
#scat rubygems-1.3.1.tgz | tar -xzf -
cd rubygems-1.3.1
ruby setup.rb
gem --version
```

Installer puppet avec la commande et l'argument -p si vous avez un proxy :

```
gem
```

```
gem install puppet --version '0.25.4' -p http://proxy:3128/
```

Il faut modifier/ajouter quelques commandes qui ne sont pas par défaut sur Solaris, pour que Puppet fonctionne mieux:

- Créer un lien pour uname et puppetd :

```
ln
ln -s /usr/bin/uname /usr/bin/
ln -s /opt/csw/bin/puppetd /usr/bin/
```

- Créer un script appelé /usr/bin/dnsdomainname :

```
/usr/bin/dnsdomainname
0. #!/usr/bin/bash
1. DOMAIN=""
2. if [ ! -z "$DOMAIN" ]; then
3.     echo $DOMAIN | sed 's/^[\*\.]*/'
4. fi
```

```
chmod
chmod 755 /usr/bin/dnsdomainname
```

Ensuite, la procédure est la même que pour les autres OS, c'est à dire, modifier /etc/hosts pour inclure puppet-prd.deimos.fr, et lancer:

```
puppetd
puppetd --verbose --no-daemon --test --server puppet-prd.deimos.fr
```

A ce stade là, si ça ne fonctionne pas, c'est tout simplement qu'il faut modifier la configuration (puppet.conf) de votre client.

4 Configuration

4.1 Serveur

Pour la partie serveur, voici comment est constitué l'arborescence de celui ci (dans /etc/puppet) :

```
0. .
1. |-- auth.conf
2. |-- autosign.conf
3. |-- fileserver.conf
4. |-- manifests
5. |   |-- common.pp
6. |   |-- modules.pp
7. |   |-- site.pp
8. |-- modules
9. |-- puppet.conf
10. |-- templates
```

4.1.1 auth.conf

C'est ici que nous allons régler toutes les autorisations :

```
/etc/puppet/auth.conf
0. # This is an example auth.conf file, it mimics the puppetmasterd defaults
1. #
2. # The ACL are checked in order of appearance in this file.
3. #
4. # Supported syntax:
5. # This file supports two different syntax depending on how
6. # you want to express the ACL.
7. #
8. # Path syntax (the one used below):
9. # -----
10. # path /path/to/resource
11. # [environment envlist]
12. # [method methodlist]
13. # [auth[en|t|ic|a|t|e|d]] {yes|no|on|off|any}
14. # allow [host|ip]*
15. # deny [host|ip]
16. #
17. # The path is matched as a prefix. That is /file match at
18. # the same time /file_metadat and /file_content.
19. #
20. # Regex syntax:
21. # -----
22. # This one is differentiated from the path one by a '-'
23. #
24. # path - regex
25. # [environment envlist]
26. # [method methodlist]
27. # [auth[en|t|ic|a|t|e|d]] {yes|no|on|off|any}
28. # allow [host|ip]*
29. # deny [host|ip]
```



```

30. #
31. # The regex syntax is the same as ruby ones.
32. #
33. # Ex:
34. # path - .pp$
35. # will match every resource ending in .pp (manifests files for instance)
36. #
37. # path - ~/path/to/resource
38. # is essentially equivalent to path /path/to/resource
39. #
40. # environment:: restrict an ACL to a specific set of environments
41. # method:: restrict an ACL to a specific set of methods
42. # auth:: restrict an ACL to an authenticated or unauthenticated request
43. # the default when unspecified is to restrict the ACL to authenticated requests
44. # (ie exactly as if auth yes was present).
45. #
46.
47. ### Authenticated ACL - those applies only when the client
48. ### has a valid certificate and is thus authenticated
49.
50. # allow nodes to retrieve their own catalog (ie their configuration)
51. path - ~/catalog/([^/]+)$
52. method find
53. allow $1
54.
55. # allow nodes to retrieve their own node definition
56. path ~/node/([^/]+)$
57. method find
58. allow $1
59.
60. # allow all nodes to access the certificates services
61. path /certificate_revocation_list/ca
62. method find
63. allow *
64.
65. # allow all nodes to store their reports
66. path /report
67. method save
68. allow *
69.
70. # inconditionally allow access to all files services
71. # which means in practice that fileserv.conf will
72. # still be used
73. path /file
74. allow *
75.
76. ### Unauthenticated ACL, for clients for which the current master doesn't
77. ### have a valid certificate; we allow authenticated users, too, because
78. ### there isn't a great harm in letting that request through.
79.
80. # allow access to the master CA
81. path /certificate/ca
82. method find
83. allow *
84.
85. path /certificate/
86. method find
87. allow *
88.
89. path /certificate_request
90. method find, save
91. allow *
92.
93. # this one is not stricly necessary, but it has the merit
94. # to show the default policy which is deny everything else
95. path /
96. auth any
97. allow *.deimos.fr

```

Dans le cas ou vous rencontrez des problèmes d'accès, pour faire simple **mais non sécurisé**, ajoutez cette ligne à la fin de votre fichier de configuration le temps de vos tests :

 /etc/puppet/auth.conf

```
0. allow *
```

4.1.2 autosign.conf

Vous pouvez autosigner certains certificats pour gagner du temps. Ceci peut être un peu dangereux, mais si votre filtrage des noeuds est correctement faite derrière, pas de soucis :-)

 autosign.conf

```
0. *.deimos.fr
```

Ici je vais autosigner tous mes nodes ayant comme domaine deimos.fr

4.1.3 fileserv.conf

Donnez les autorisations des machines clientes dans le fichier /etc/puppet/fileserv.conf :

 /etc/puppet/fileserv.conf

```

0. # This file consists of arbitrarily named sections/modules
1. # defining where files are served from and to whom
2.
3. # Define a section 'files'
4. # Adapt the allow/deny settings to your needs. Order
5. # for allow/deny does not matter, allow always takes precedence
6. # over deny
7. [files]
8. path /etc/puppet/files
9. allow *.deimos.fr
10. # allow *.example.com
11. # deny *.evil.example.com
12. # allow 192.168.0.0/24
13.
14. [plugins]

```

```
15. allow *.deimos.fr
16. # allow *.example.com
17. # deny *.evil.example.com
18. # allow 192.168.0.0/24
```

4.1.4 manifests

Créons les fichiers manquants :

 touch

```
0. touch /etc/puppet/manifests/{common.pp,modules.pp,site.pp}
```

4.1.4.1 common.pp


Puis on va les renseigner un par un. Le common.pp est vide, mais vous pouvez y insérer des choses qui permettront d'être pris en tant que configuration globale.

 /etc/puppet/manifests/common.pp

```
0.
```

4.1.4.2 modules.pp


Ensuite je vais définir ici mon ou mes modules de base. Par exemple, dans ma future configuration je vais déclarer un module "base" qui contiendra tout ce que n'importe quelle machine faisant partie de puppet héritera :

 /etc/puppet/manifests/modules.pp

```
0. import "base"
```

4.1.4.3 site.pp

On demande de charger tous les modules présents dans le dossier modules :

 /etc/puppet/manifests/site.pp

```
0. # /etc/puppet/manifests/site.pp
1. import "common.pp"
2. import "modules.pp"
3.
4. # The filebucket option allows for file backups to the server
5. filebucket { main: server => 'puppet-prod-nux.deimos.fr' }
6.
7. # Backing up all files and ignore vcs files/folders
8. File {
9.     backup => ['.puppet-bak'],
10.    ignore => ['.svn', '.git', 'CVS' ]
11. }
12.
13. # Default global path
14. Exec { path => "/usr/bin:/usr/sbin:/bin:/sbin" }
```

Ici je lui dis d'utiliser le filebucket présent sur le serveur puppet et de renommer les fichiers qui vont être remplacés par puppet en <fichier>.puppet-bak.

Je lui demande également d'ignorer tout dossiers ou fichiers créés par des VCS de type SVN, git ou CVS.

Et enfin, j'indique le path par défaut que puppet aura quand il s'exécutera sur les clients.


Il faut savoir que toute cette configuration est propre au serveur puppet, donc global. Tous ce que nous mettrons dedans pourra être hérité.

On relance le puppetmaster pour être sûr que les modifications côté serveur ont bien été prises en compte.

4.1.5 puppet.conf

J'ai volontairement passé le dossier modules car c'est le gros morceau de puppet et fera l'objet d'une attention toute particulière plus tard dans cet article.

Donc nous allons passer au fichier de configuration puppet.conf que vous avez normalement déjà configuré lors de l'installation mongrel/nginx... :

 /etc/puppet/puppet.conf

```
0. [main]
1. logdir=/var/log/puppet
2. vardir=/var/lib/puppet
3. ssl_dir=/var/lib/puppet/ssl
4. rundir=/var/run/puppet
5. factpath=$vardir/lib/facter
6. templatedir=$confdir/templates
7.
8. [master]
9. # These are needed when the puppetmaster is run by passenger
10. # and can safely be removed if webrick is used.
11. ssl_client_header = HTTP_X_SSL_SUBJECT
```

```
12. ssl_client_verify_header = SSL_CLIENT_VERIFY
13. report = true
14.
15. [main]
16. pluginsync = true
```

Pour le dossier templates, je n'ai rien dedans.

4.2 Client

Chaque client doit avoir son entrée dans le serveur DNS (tout comme le serveur) !

4.2.1 puppet.conf

4.2.1.1 Debian / Red Hat

Le fichier de configuration doit contenir l'adresse du serveur :

```
etc/puppet/puppet.conf

0. [main]
1. # The Puppet log directory.
2. # The default value is '$vardir/log'.
3. logdir = /var/log/puppet
4.
5. # Where Puppet PID files are kept.
6. # The default value is '$vardir/run'.
7. rundir = /var/run/puppet
8.
9. # Where SSL certificates are kept.
10. # The default value is '$confdir/ssl'.
11. ssl_dir = $vardir/ssl
12.
13. # Puppet master server
14. server = puppet-prd.deimos.fr
15.
16. # Add custom facts
17. pluginsync = true
18. plugin_source = puppet://$server/plugins
19. factpath = /var/lib/puppet/lib/facter
20.
21. [agent]
22. # The file in which puppetd stores a list of the classes
23. # associated with the retrieved configuration. Can be loaded in
24. # the separate ``puppet`` executable using the ``--loadclasses``
25. # option.
26. # The default value is '$confdir/classes.txt'.
27. classfile = $vardir/classes.txt
28.
29. # Where puppetd caches the local configuration. An
30. # extension indicating the cache format is added automatically.
31. # The default value is '$confdir/localconfig'.
32. localconfig = $vardir/localconfig
33.
34. # Reporting
35. report = true
```

4.2.1.2 Solaris

Pour la partie Solaris, il a fallu pas mal adapter la configuration :

```
etc/puppet/puppet.conf

0. [main]
1. logdir=/var/log/puppet
2. vardir=/var/opt/csw/puppet
3. rundir=/var/run/puppet
4. # ssl_dir=/var/lib/puppet/ssl
5. ssl_dir=/etc/puppet/ssl
6. # Where 3rd party plugins and modules are installed
7. libdir = $vardir/lib
8. templatedir=$vardir/templates
9. # Turn plug-in synchronization on.
10. pluginsync = true
11. plugin_source = puppet://$server/plugins
12. factpath = /var/puppet/lib/facter
13.
14. [puppetd]
15. report=true
16. server=puppet-prd.deimos.fr
17. # certname=puppet-prd.deimos.fr
18. # enable the marshal config format
19. config_format=marshal
20. # different run-interval, default= 30min
21. # e.g. run puppetd every 4 hours = 14400
22. runinterval = 14400
23. logdest=/var/log/puppet/puppet.log
```

5 Le langage

Avant de commencer la création de modules, il va falloir en connaître un peu plus sur la syntaxe/le langage utilisé pour puppet. Il faut savoir que sa syntaxe est proche du ruby et qu'il est même possible d'écrire des modules complets en ruby. Je vais expliquer ici quelques techniques/possibilités pour vous permettre de créer des modules avancés par la suite.

Nous allons utiliser des types également, je ne rentrerais pas en détail dessus, car la doc sur le site est suffisamment clair pour cela : <http://docs.puppetlabs.com/references/latest/type.html>

5.1 Les fonctions

Voici comment définir une fonction avec plusieurs arguments :

```

0. define network_config( $ip, $netmask, $gateway ) {
1.   notify {"$ip, $netmask, $gateway"}
2. }
3. network_config { "eth0":
4.   ip    => '192.168.0.1',
5.   netmask => '255.255.255.0',
6.   gateway => '192.168.0.254',
7. }

```

5.2 Installer des packages

Nous allons voir ici comment installer un package, puis utiliser une fonction pour facilement en installer bien plus. Pour un package, c'est simple :

init.pp

```

0. # Install kexec-tools
1. package { 'kexec-tools':
2.   ensure => 'installed'
3. }

```

Nous demandons ici qu'un package (kexec-tool) soit installé. Si nous souhaitons que plusieurs soient installés, il va falloir créer un tableau :

init.pp

```

0. # Install kexec-tools
1. package {
2.   [
3.     'kexec-tools',
4.     'package2',
5.     'package3'
6.   ]:
7.   ensure => 'installed'
8. }

```

C'est plutôt pratique et les tableaux fonctionnent très souvent de cette manière pour un peu près n'importe quel type utilisé. Nous pouvons aussi créer une fonction pour cela dans laquelle nous allons lui envoyer chaque élément du tableau :

init.pp

```

0. # Validate that packages are installed
1. define packages_install () {
2.   notice("Installation of ${name} package")
3.   package {
4.     "${name}":
5.     ensure => 'installed'
6.   }
7. }
8. # Set all custom packages (not embedded in distribution) that need to be installed
9. packages_install
10. { [
11.   'puppet',
12.   'tmux'
13. ]: }

```

Certains d'entre vous diront que pour ce cas précis, ça ne sert à rien, puisque la méthode au dessus permet de le faire tandis que d'autres trouveront cette méthode plus élégante et facile à appréhender pour un novice qui arrive sur puppet. Le nom de la fonction utilisée est packages_install, la variable \$name est toujours le premier élément envoyé à une fonction, qui correspond ici à chaque élément de notre tableau.

5.3 Inclure / Exclure des modules

Vous venez de voir les fonctions, nous allons les pousser un peu plus loin avec une solution pour inclure et exclure le chargement de certains modules. Ici, j'ai un fichier de fonctions :

functions.pp

```

0. # Load or not modules (include/exclude)
1. define include_modules () {
2.   if ($exclude_modules == undef) or !($name in $exclude_modules) {
3.     include $name
4.   }
5. }

```

Là j'ai un autre fichier représentant les rôles de mes serveurs (on y viendra plus tard) :

init.pp

```

0. # Load modules
1. $minimal_modules =
2. [
3.   'puppet',
4.   'resolvconf',
5.   'packages_defaults',
6.   'configurations_defaults'
7. ]
8. include_modules{ $minimal_modules: }

```

Et enfin un fichier contenant le nom d'un serveur dans lequel je vais lui demander de charger certains modules, mais d'en exclure également une partie :

```
0. node 'srv.deimos.fr' {
  1.   $exclude_modules = [ 'resolvconf' ]
  2.   include base::minimal
  3. }
```

Ici j'utilise un tableau '\$exclude_modules' (avec un seul élément, mais vous pouvez en mettre plusieurs séparés par des virgules), qui va me permettre de préciser les modules à exclure. Car par la ligne d'après il va charger tout ce dont il aura besoin grâce à la fonction include_modules.

5.4 Les templates

Lorsque vous écrivez des manifests, vous faites appelle à une directive nommée 'File' lorsque vous souhaitez envoyer un fichier sur un serveur. Mais si le contenu de ce fichier doit changer en fonction de certains paramètres (nom, ip, timezone, domaine...), alors il faut utiliser les templates ! Et c'est là que ça devient intéressant puisqu'il est possible de scripter au sein même d'un template pour en générer le contenu. Les templates utilisent un langage très proche de celui du ruby.

Voici un exemple avec OpenSSH pour que vous compreniez. J'ai donc pris la configuration qui va varier selon certains paramètres :

```
etc/puppet/modules/ssh/templates/sshd_config

# Package generated configuration file
# See the sshd(8) manpage for details
#
# What ports, IPs and protocols we listen for
<%= ssh_default_port.each do |val| -%>
Port <%= val -%>
<% end -%>
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
#Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes

#RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile  %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

UsePAM yes
# AllowUsers <%= ssh_allowed_users %>
```

Ici nous utilisons donc 2 types d'utilisation de templates. Une multi lignes a répétition, et l'autre avec un simple remplacement de variables :

- ssh_default_port.each do : permet de mettre une ligne de "Port num_port" a chaque port spécifié
- ssh_allowed_users : permet de donner une liste d'utilisateur

Ces variables sont généralement a déclarer soit dans la partie 'node' ou bien dans la configuration globale. Nous venons de voir comment mettre une variable ou une boucle dans un template, mais sachez qu'il est également possible de mettre des if ! Bref, un langage complet existe et vous permet de moduler a souhait un fichier.

Ces méthodes s'avèrent simple et très efficaces. Petite subtilité :

- %> : Lorsqu'une ligne se termine comme ceci, c'est qu'il ne va pas y avoir de saut de ligne grâce au - situé à la fin.
- %> : Il y aura un saut de ligne ici.

5.4.1 Les inline-templates

Ceci est une petite subtilité qui peut paraître inutile, mais est en fait très utile pour exécuter de petites méthodes au sein d'un manifest ! Prenez par exemple la fonction 'split' qui aujourd'hui existe dans puppet, il semblerait normal que la fonction 'join' existe non ? Et bien non...enfin pas dans la version actuelle au moment où j'écris ces lignes (la 2.7.18). Je peux donc utiliser de la même façon que les templates du code dans mes manifests, voyez plutôt :

```
0. $ldap_servers = [ '192.168.0.1', '192.168.0.2', '127.0.0.1' ]
1. $comma_ldap_servers = inline_template("<%= (ldap_servers).join(',') %>")
```

- \$ldap_servers : ceci est un simple tableau avec ma liste de serveurs LDAP
- \$comma_ldap_servers : nous utilisons la fonction inline_template, qui va appeler la fonction join, lui passer le tableau ldap_servers et joindre le contenu avec des virgules.

J'aurais au finale :

```
0. $comma_ldap_servers = '192.168.0.1,192.168.0.2,127.0.0.1'
```

5.5 Les Factors

Les "facts", sont des scripts (voir /usr/lib/ruby/1.8/facter pour les facts standards) permettant de construire des variables dynamiques, qui changent en fonction de l'environnement dans lequel ils sont exécutés.

Par exemple, on pourra définir un "fact", qui détermine si l'on est sur une machine de type "cluster" en fonction de la présence ou l'absence d'un fichier :

is_cluster.rb

```
0. # is_cluster.rb
1.
2. Facter.add("is_cluster") do
3.   setcode do
4.     FileTest.exists?("/etc/cluster/nodeid")
5.   end
6. end
```

On peut aussi utiliser de fonctions qui permettent de mettre directement dans des templates, des fonctions de type factor (downcase ou upcase pour changer la casse) :

modules/collectd/templates/collectd.conf

```
0. #
1. # Config file for collectd(1).
2. # Please read collectd.conf(5) for a list of options.
3. # http://collectd.org/
4. #
5.
6. Hostname      <%= hostname.downcase %>
7. FQDNLookup    true
```

Attention, si l'on veut tester le fact sur la machine destination, il ne faudra pas oublier de spécifier le chemin où se trouvent les facts sur la machine :

export

```
export FACTERLIB=/var/lib/puppet/lib/facter
```

ou pour Solaris :

export

```
export FACTERLIB=/var/opt/csw/puppet/lib/facter
```

Pour voir la liste des facts ensuite présent sur le système, il faut simplement taper la commande facter :

facter

```
0. > facter
1. facterversion => 1.5.7
2. hardwareisa => i386
3. hardwaremodel => i86pc
4. hostname => PA-OFC-SRV-UAT-2
5. hostnameldap => PA-OFC-SRV-UAT
6. id => root
7. interfaces => lo0,e1000g0,e1000g0_1,e1000g0_2,e1000g1,e1000g2,e1000g3,clprivnet0
8. ...
```

Voir http://docs.puppetlabs.com/guides/custom_facts.html pour plus de détails.

5.6 Informations dynamiques

Il est possible d'utiliser des scripts côté serveur et d'en récupérer le contenu dans une variable. Voici un exemple :

/usr/bin/latest_puppet_version.rb

```

0. #!/usr/bin/ruby
1. require 'open-uri'
2. page = open("http://www.puppetlabs.com/misc/download-options/").read
3. print page.match(/stable version is ([\d\.]*)/)[1]

```

Et dans le manifest :

```

0. $latestversion = generate("/usr/bin/latest_puppet_version.rb")
1. notify { "The latest stable Puppet version is ${latestversion}. You're using ${puppetversion}." }

```

Magique non ? :-). Sachez qu'il est même possible de passer des arguments avec une virgule entre chaque !!!

5.7 Les Parsers

Les parsers sont la création de fonctions particulières utilisables dans les manifests (côté serveur). Par exemple, j'ai créé un parser qui va me permettre de faire du reverse lookup dns :

 /etc/puppet/modules/openldap/lib/puppet/parser/functions/dns2ip.rb

```

0. # Dns2IP for Puppet
1. # Made by Pierre Mavro
2. # Does a DNS lookup and returns an array of strings of the results
3. # Usage : need to send one string dns servers separated by comma. The return will be the same
4.
5. require 'resolv'
6.
7. module Puppet::Parser::Functions
8.   newfunction(:dns2ip, :type => :rvalue) do |arguments|
9.     result = [ ]
10.    # Split comma sperated list in array
11.    dns_array = arguments[0].split(',')
12.    # Push each DNS/IP address in result array
13.    dns_array.each do |dns_name|
14.      result.push(Resolv.new.getaddresses(dns_name))
15.    end
16.    # Join array with comma
17.    dns_list = result.join(',')
18.    # Delete last comma if exist
19.    good_dns_list = dns_list.gsub(/,$/, '')
20.    return good_dns_list
21.  end
22. end

```

Nous allons pouvoir créer cette variable puis l'insérer dans nos manifests :

```

0. $comma_ldap_servers = 'ldap1.deimos.fr,ldap2.deimos.fr,127.0.0.1'
1. $ip_ldap_servers = dns2ip("${comma_ldap_servers}")

```

J'envoie ici une liste de serveur LDAP et il me sera retourné leurs adresses IP. Vous comprenez maintenant que c'est un appel, un peu comme les inline_templates, mais qui est bien plus puissant.

 Notes

J'ai pu constater un comportement de cache assez désagréable avec ce type de fonctions ! En effet, lorsque vous développez un parser et que vous le testez, il est fort probable que vous utilisiez des fonctions 'Notify' dans vos manifests pour debugger. Cependant les modifications que vous ferez sur votre parser ne s'appliqueront pas forcément tant que vous n'aurez pas vidé les caches. Après quelques recherches et demandes IRC, il s'avère que la seule méthode fonctionnelle soit de **redémarrer le Puppet Master et le serveur web (Nginx dans notre cas)**. Ça fonctionne très bien, mais c'est un peu pénible lors de la phase de debug.

5.8 Du Ruby dans vos manifests

Il est tout à fait possible d'écrire du Ruby dans vos manifests. Voyez plutôt :

```

0. notice( "I am running on node %s" % scope.lookupvar("fqdn") )

```

Ca ressemble fortement à un sprintf.

5.9 Ajouter une variable Ruby dans les manifests

Si nous souhaitons par exemple récupérer l'heure actuelle dans un manifest :

```

0. require 'time'
1. scope.setvar("now", Time.now)
2. notice( "Here is the current time : %s" % scope.lookupvar("now") )

```

5.10 Les classes

On peut utiliser des classes avec des arguments comme ceci :

```

0. class mysql( $package, $socket, $port = "3306" ) {
1.   ..

```

```

2. }
3. class { "mysql":
4.   package => "percona-sql-server-5.0",
5.   socket => "/var/run/mysqlid/mysqlid.sock",
6.   port => "3306",
7. }

```

5.11 Utilisation des tables de hash

Tout comme les tableaux, il est également possible d'utiliser les tables de hash, regardez cet exemple :

```

0. $interface = {
1.   name => 'eth0',
2.   address => '192.168.0.1'
3. }
4. notice("Interface ${interface[name]} has address ${interface[address]}")

```

5.12 Les regex

Il est possible d'utiliser les regex et d'en récupérer les patterns :

```

0. $input = "What a great tool"
1. if $input =~ /What a (\w+) tool/ {
2.   notice("You said the tool is : '$1'. The complete line is : $0")
3. }

```

5.13 Substitution

Il est possible de substituer :

```

0. $ipaddress = '192.168.0.15'
1. $class_c = regexsubst($ipaddress, "(.*)\.\.+", "\1.0")
2. notify { $ipaddress: }
3. notify { $class_c: }

```

Ce qui me donnera 192.168.0.15 et 192.168.0.0.

5.14 Notify et Require

Ces 2 fonctions sont fortes utiles une fois insérées dans un manifest. Cela permet par exemple à un service de dire qu'il require (require) un Package pour fonctionner et à un fichier de configuration de notifier (notify) un service s'il change pour que celui ci redémarre le démon. On peut également écrire quelque chose comme ceci :

```

0. Package["ntp"] -> File["/etc/ntp.conf"] -> Service["ntp"]

```

- ->: signifie 'require'
- ~>: signifie 'notify'

Il est également possible de faire des require sur des classes :-)

5.15 L'opérateur +>

Voici un superbe opérateur qui va nous permettre de gagner un peu de temps. L'exemple ci dessous :

```

0. file { "/etc/ssl/certs/cookbook.pem":
1.   source => "puppet:///modules/apache/deimos.pem",
2. }
3. Service["apache2"] {
4.   require +> File["/etc/ssl/certs/deimos.pem"],
5. }

```

Correspond à :

```

0. service { "apache2":
1.   enable => true,
2.   ensure => running,
3.   require => File["/etc/ssl/certs/deimos.pem"],
4. }

```

5.16 Vérifier le numéro de version d'un soft

Si vous avez besoin de vérifier le numéro e version d'un soft pour prendre ensuite une décision, voici un bon exemple :

```

0. $app_version = "2.7.16"
1. $min_version = "2.7.18"
2. if versioncmp( $app_version, $min_version ) >= 0 {
3.   notify { "Puppet version OK": }
4. } else {
5.   notify { "Puppet upgrade needed": }
6. }

```

5.17 Les ressources virtuelles

Utile pour les écritures de test, vous pouvez par exemple créer une ressource en la précédant d'un '@'. Elle sera lue mais non exécutée jusqu'à ce qu'implicitement vous lui indiquiez (realize). Exemple :


```

0. @package {
1.   'postfix':
2.     ensure => installed
3. }
4. realize( Package['postfix'] )

```

Un des gros avantages de cette méthode est de pouvoir déclarer à plusieurs endroits dans votre puppet master le realize sans pour autant que vous ayez de conflits !

5.18 Suppression d'un fichier évoluée

Vous pouvez demander la suppression d'un fichier au bout d'un temps donné ou bien à partir d'une certaine taille :

```

0. tidy { "/var/lib/puppet/reports":
1.   age => "1w",
2.   size => "512K",
3.   recurse => true,
4. }

```

Ceci entrainera la suppression d'un dossier au bout d'une semaine avec son contenu.

6 Les modules

Il est recommandé de créer des modules pour chaque service afin de rendre la configuration plus souple. Ca fait partie de certaines best practices.

Je vais aborder ici différentes techniques en essayant de garder un ordre croissant de difficulté.

6.1 Initialisation d'un module

Nous allons donc créer sur le serveur, l'arborescence adéquate. Pour cet exemple, nous allons partir avec "sudo", mais vous pouvez choisir autre chose si vous souhaitez :

```

0. mkdir -p /etc/puppet/modules/sudo/manifests
1. touch /etc/puppet/modules/sudo/manifests/init.pp

```

N'oubliez pas que ceci est nécessaire pour chaque module. Le fichier init.pp est le premier fichier qui se chargera lors de l'appel au module.

6.2 Le module initiale (base)

Il nous faut créer un module initiale qui va gérer la liste des serveurs, les fonctions dont nous allons avoir besoin, les roles, des variables globales... bref, ça peut paraître un peu abstrait au premier abord mais sachez juste qu'il nous faut un module pour gérer ensuite tous les autres. Nous allons commencer par celui là qui est un des plus important pour la suite.

Comme vous le savez maintenant, il nous faut un fichier init.pp pour que le premier module soit chargé. Nous allons donc créer notre arborescence que nous allons appeler "base" :

```

0. mkdir -p /etc/puppet/modules/base/{manifests,puppet/parser/functions}

```

6.2.1 init.pp

Puis nous allons créer et renseigner le fichier init.pp :

```

/etc/puppet/modules/base/manifests/init.pp
0. #####
1. # BASE MODULES #
2. #####
3.
4. # Load defaults vars
5. import "vars.pp"
6. # Load functions
7. import "functions.pp"
8. # Load sysctl module
9. include "sysctl"
10. # Load network module
11. include "network"
12. # Load roles
13. import "roles.pp"
14. # Set servers properties
15. import "servers.pp"

```

Les lignes correspondantes à import sont équivalentes à un "include" (dans des services comme ssh ou ntp) de mes autres fichier .pp que nous allons créer par la suite. Tandis que les include vont charger d'autres modules que je vais créer par la suite.

6.2.2 vars.pp

Nous allons ensuite créer le fichier vars.pp qui contiendra toutes mes variables globales pour mes futurs modules ou manifests (*.pp) :

```

/etc/puppet/modules/base/manifests/vars.pp

```

```

0. #####
1. # VARS #
2. #####
3.
4. # Default admins emails
5. $root_email = 'deimos@deimos.fr'
6.
7. # NTP Timezone. Usage :
8. # Look at /usr/share/zoneinfo/ and add the continent folder followed by the town
9. $set_timezone = 'Europe/Paris'
10.
11. # Define empty exclude modules
12. $exclude_modules = [ ]
13.
14. # Default LDAP servers
15. $ldap_servers = [ ]
16.
17. # Default DNS servers
18. $dns_servers = [ '192.168.0.69', '192.168.0.27' ]

```

6.2.3 functions.pp

Maintenant, nous allons créer des fonctions qui vont nous permettre de rajouter quelques fonctionnalités aujourd'hui non présentes dans puppet ou en simplifier certaines :

```

/etc/puppet/modules/base/manifests/functions.pp

0. /*
1. Puppet Functions
2. Made by Pierre Mavro
3. */
4. #####
5. # GLOBAL FUNCTIONS #
6. #####
7. /*
8.
9. # Load or not modules (include/exclude)
10. define include_modules () {
11.   if ($exclude_modules == undef) or !($name in $exclude_modules) {
12.     include $name
13.   }
14. }
15.
16. # Validate that packages are installed
17. define packages_install () {
18.   notice("Installation of ${name} package")
19.   package {
20.     "${name}":
21.       ensure => present
22.   }
23. }
24.
25. # Check that those services are enabled on boot or not
26. define services_start_on_boot ($enable_status) {
27.   service {
28.     "${name}":
29.       enable => "${enable_status}"
30.   }
31. }
32.
33. # Add, remove, comment or uncomment lines
34. define line ($file, $line, $ensure = 'present') {
35.   case $ensure {
36.     default : {
37.       err("unknown ensure value ${ensure}")
38.     }
39.     present : {
40.       exec {
41.         "echo '${line}' >> '${file}":
42.           unless => "grep -qFx '${line}' '${file}";
43.           logoutput => true
44.       }
45.     }
46.     absent : {
47.       exec {
48.         "grep -vFx '${line}' '${file}' | tee '${file}' > /dev/null 2>&1":
49.           onlyif => "grep -qFx '${line}' '${file}";
50.           logoutput => true
51.       }
52.     }
53.     uncomment : {
54.       exec {
55.         "sed -i -e '/${line}/s/#\+//' '${file}":
56.           onlyif => "test `grep '${line}' '${file}' | grep '^#' | wc -l` -ne 0";
57.           logoutput => true
58.       }
59.     }
60.     comment : {
61.       exec {
62.         "/bin/sed -i -e '/${line}/s/(.+\+)/#\1/' '${file}":
63.           onlyif => "test `grep '${line}' '${file}' | grep -v '^#' | wc -l` -ne 0";
64.           logoutput => true
65.       }
66.     }
67.   }
68.   # Use this resource instead if your platform's grep doesn't support -vFx;
69.   # note that this command has been known to have problems with lines containing quotes.
70.   # exec { "/usr/bin/perl -ni -e 'print unless /\Q${line}\E\/' '${file}":
71.   #   onlyif => "grep -qFx '${line}' '${file}";
72.   # }
73. }
74. }
75. }
76.
77. # Validate that softwares are installed
78. define comment_lines ($filename) {
79.   line {
80.     "${name}":
81.       file => "${filename}",
82.       line => "${name}",
83.       ensure => comment
84.   }
85. }
86.
87. # Sysctl managment
88. class sysctl {
89.   define conf ($value) {
90.     # $name is provided by define invocation
91.     # guild of this entry
92.     $key = $name
93.     $context = "/files/etc/sysctl.conf"
94.     augeas {
95.       "sysctl_conf/$key":
96.         context => "$context",
97.         onlyif => "get $key != '$value'",

```

```

98.         changes => "set $key '$value'",
99.         notify => Exec["sysctl"],
100.     }
101. }
102. file {
103.     "sysctl.conf" :
104.         name => $::operatingsystem ? {
105.             default => "/etc/sysctl.conf",
106.         },
107.     }
108.     exec {
109.         "sysctl -p" :
110.             alias => "sysctl",
111.             refreshonly => true,
112.             subscribe => File["sysctl.conf"],
113.         }
114. }
115.
116. # Function to add ssh public keys
117. define ssh_add_key($user, $key) {
118.     # Create users home directory if absent
119.     exec {
120.         "mkhomedir_${name}" :
121.             path => "/bin:/usr/bin",
122.             command => "cp -Rfp /etc/skel -${user}; chown -Rf ${user}:group -${user}",
123.             onlyif => "test `ls -${user} 2>&1 >/dev/null | wc -l` -ne 0"
124.         }
125.     }
126.     ssh_authorized_key {
127.         "${name}" :
128.             ensure => present,
129.             key => "$key",
130.             type => 'ssh-rsa',
131.             user => "$user",
132.             require => Exec["mkhomedir_${name}"]
133.         }
134.     }
135. }
136.
137. # Limits.conf management
138. define limits_conf($domain = "root", $type = "soft", $item = "nofile", $value = "10000") {
139.     # guid of this entry
140.     $key = "$domain/$type/$item"
141.     # augtool> match /files/etc/security/limits.conf/domain[.="root"]/.type="hard" and ./item="nofile" and ./value="10000"
142.     $context = "/files/etc/security/limits.conf"
143.     $path_list = "domain[.="${domain}"]/.type=${type}" and ./item=${item}"
144.     $path_exact = "domain[.="${domain}"]/.type=${type}" and ./item=${item}" and ./value=${value}"
145.
146.     augears {
147.         "limits_conf/$key" :
148.             context => "$context",
149.             onlyif => "match $path_exact size==0",
150.             changes => [
151.                 # remove all matching to the $domain, $type, $item, for any $value
152.                 "rm $path_list",
153.                 # insert new node at the end of tree
154.                 "set domain[last()+1] $domain",
155.                 # assign values to the new node
156.                 "set domain[last()]/type $type",
157.                 "set domain[last()]/item $item",
158.                 "set domain[last()]/value $value",],
159.             }
160.     }

```

Nous avons donc :

- Ligne 10 : La possibilité de charger ou non des modules via un tableau envoyé en argument de fonction (comme décrit plus haut dans cette documentation)
- Ligne 17 : La possibilité de vérifier que des packages sont installés sur la machine
- Ligne 26 : La possibilité de vérifier que des services sont correctement chargés au boot de la machine
- Ligne 34 : La possibilité de s'assurer qu'une ligne d'un fichier est présente, absente, commentée ou non non commentée
- Ligne 78 : La possibilité de commenter plusieurs lignes via un tableau envoyé en argument de fonction
- Ligne 88 : La possibilité de gérer le fichier sysctl.conf
- Ligne 117 : La possibilité de déployer facilement des clés publiques SSH
- Ligne 128 : La possibilité de gérer simplement le fichier limits.conf

Toutes ses fonctions ne sont bien entendues pas obligatoires mais aident grandement à l'utilisation de puppet.

6.2.4 roles.pp

Ensuite nous avons un fichier contenant les rôles des serveurs. Voyez ça plutôt comme des groupes auxquels nous allons faire souscrire les serveurs :

```

/etc/puppet/modules/base/manifests/roles.pp
1. #####
2. # ROLES
3. #####
4. # Level 1 : Minimal
5. class base::minimal
6. {
7.     # Load modules
8.     $minimal_modules =
9.     [
10.         'puppet',
11.         'resolvconf',
12.         'packages_defaults',
13.         'configurations_defaults',
14.         'openssh',
15.         'selinux',
16.         'grub',
17.         'kdump',
18.         'tools',
19.         'timezone',
20.         'ntp',
21.         'mysecureshell',
22.         'openldap',
23.         'acl',
24.         'sudo',
25.         'snmpd',
26.         'postfix',
27.         'nrpe'
28.     ]
29.     include_modules{ $minimal_modules: }
30. }
31.
32. # Level 2 : Cluster
33. class base::cluster inherits minimal
34. {
35.     # Load modules
36.     $cluster_modules =
37.     [

```

```

38.     'packages_cluster',
39.     'configurations_cluster'
40.   ]
41.   include_modules{ $cluster_modules: }
42. }
43.
44. # Level 2 : Low Latency
45. class base::low_latency inherits minimal
46. {
47.   # Load modules
48.   $lowlatency_modules =
49.   [
50.     'low_latency'
51.   ]
52.   include_modules{ $lowlatency_modules: }
53. }
54.
55. # Level 3 : Low Latency + Cluster
56. class base::low_latency_cluster inherits minimal
57. {
58.   include base::cluster
59.   include base::low_latency
60. }

```

J'ai donc défini ici des classes qui héritent plus ou moins entre elles. C'est en fait défini par niveaux. Le niveau 3 dépend du 2 et 1, le 2 du 1 et le 1 n'a pas de dépendances. Cela me permet d'avoir une certaine souplesse. Je sais par exemple ici que si je charge ma classe cluster, ma classe minimal sera également chargée. Vous noterez l'annotation 'base::minimal'. Il est recommandé de charger ses classes en appelant le module, suivi de '::'. Cela facilite grandement la lecture des manifests.

6.2.5 servers.pp

Et pour finir, j'ai un fichier où je fais ma déclaration de serveurs :

```

/etc/puppet/modules/base/manifests/servers.pp
0. /*#####
1. #
2. # SERVERS
3. #
4. == Automated Dependencies Roles ==
5. * cluster -> minimal
6. * low_latency -> minimal
7. * low_latency_cluster -> low_latency + cluster + minimal
8.
9. == Template for servers ==
10. node /regex/
11. {
12.   # $exclude_modules = [ ]
13.   # $ldap_servers = 'x.x.x.x'
14.   # $set_timezone = 'Europe/Paris'
15.   # $dns_servers = [ ]
16.   # include base::minimal
17.   # include base::cluster
18.   # include base::low_latency
19.   # include base::low_latency_cluster
20. }
21.
22. #####*/
23.
24. # One server
25. node 'srv1.deimos.fr' {
26.   $ldap_servers = [ '127.0.0.1' ]
27.   include base::minimal
28. }
29.
30. # Multiple servers
31. node 'srv2.deimos.fr' 'srv3.deimos.fr' {
32.   $ldap_servers = [ '127.0.0.1' ]
33.   include base::minimal
34. }
35.
36. # Multiple regex based servers
37. node /srv-prd-\d+/ {
38.   include base::minimal
39.   include base::low_latency
40.   $set_timezone = 'Europe/London'
41. }

```

Ici j'ai mis un serveur en exemple ou une regex pour plusieurs serveurs. Pour info, la configuration peut être intégrée dans LDAP (<http://reductivelabs.com/trac/puppet/wiki/LDAPNodes>) .

6.2.6 Parser

Nous allons créer l'arborescence nécessaire :

```

mkdir
0. mkdir -p /etc/puppet/modules/base/puppet/parser/functions

```

Puis ajoutez un parser empty qui nous permettra de détecter si un tableau/une variable est vide ou non :

```

/etc/puppet/modules/base/puppet/parser/functions/empty.rb
0. #
1. # empty.rb
2. #
3. # Copyright 2011 Puppet Labs Inc.
4. # Copyright 2011 Krzysztof Wilczynski
5. #
6. # Licensed under the Apache License, Version 2.0 (the "License");
7. # you may not use this file except in compliance with the License.
8. # You may obtain a copy of the License at
9. #
10. # http://www.apache.org/licenses/LICENSE-2.0
11. #
12. # Unless required by applicable law or agreed to in writing, software

```

```

13. # distributed under the License is distributed on an "AS IS" BASIS,
14. # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15. # See the License for the specific language governing permissions and
16. # limitations under the License.
17. #
18.
19. module Puppet::Parser::Functions
20.   newfunction(:empty, :type => :rvalue, :doc => <<-EOS
21. Returns true if given array type or hash type has no elements or when a string
22. value is empty and false otherwise.
23.
24. Prototype:
25.
26. empty(x)
27.
28. Where x is either an array, a hash or a string value.
29.
30. For example:
31.
32. Given the following statements:
33.
34. $a = ''
35. $b = 'abc'
36. $c = []
37. $d = ['d', 'e', 'f']
38. $e = {}
39. $f = { 'x' => 1, 'y' => 2, 'z' => 3 }
40.
41. notice empty($a)
42. notice empty($b)
43. notice empty($c)
44. notice empty($d)
45. notice empty($e)
46. notice empty($f)
47.
48. The result will be as follows:
49.
50. notice: Scope(Class[main]): true
51. notice: Scope(Class[main]): false
52. notice: Scope(Class[main]): true
53. notice: Scope(Class[main]): false
54. notice: Scope(Class[main]): true
55. notice: Scope(Class[main]): false
56. EOS
57. ) do [*arguments]
58.   #
59.   # This is to ensure that whenever we call this function from within
60.   # the Puppet manifest or alternatively form a template it will always
61.   # do the right thing ...
62.   #
63.   arguments = arguments.shift if arguments.first.is_a?(Array)
64.
65.   raise Puppet::ParseError, "empty(): Wrong number of arguments " +
66.     "given #{arguments.size} for 1)" if arguments.size < 1
67.
68.   value = arguments.shift
69.
70.   unless [Array, Hash, String].include?(value.class)
71.     raise Puppet::ParseError, 'empty(): Requires either array, hash ' +
72.       'or string type to work with'
73.   end
74.
75.   value.empty?
76. end
77. end
78.
79. # vim: set ts=2 sw=2 et :
80. # encoding: utf-8

```

7 Exemples de modules

7.1 Puppet

Celui ci est assez drôle car en fait il s'agit simplement de la configuration du client Puppet. Cependant, il peut s'avérer très utile pour sa gérer ses propres mises à jour. Nous allons donc créer les arborescences :



mkdir

```
0. mkdir -p /etc/puppet/modules/puppet/{manifests,files}
```

7.1.1 init.pp

Nous créons ici le module init.pp qui va nous permettre selon l'OS de choisir le fichier à charger.



/etc/puppet/modules/puppet/manifests/init.pp

```

0. /*
1. Puppet Module for Puppet
2. Made by Pierre Mavro
3. */
4. class puppet {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'RedHat' : {
8.       include ::puppet::redhat
9.     }
10.    'sunos' : { include packages_defaults::solaris }
11.    default : {
12.      notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.    }
14.  }
15. }

```

7.1.2 redhat.pp

```

/etc/puppet/modules/puppet/manifests/redhat.pp

0. /*
1. Puppet Module for Puppet
2. Made by Pierre Mavro
3. */
4. class puppet::redhat {
5.   # Change default configuration
6.   file {
7.     '/etc/puppet/puppet.conf' :
8.       ensure => present,
9.       source => "puppet:///modules/puppet/${::osfamily}.puppet.conf",
10.      mode => 644,
11.      owner => root,
12.      group => root
13.    }
14.
15.    # Disable service on boot and be sure it is not started
16.    service {
17.      'puppet-srv' :
18.        name => 'puppet',
19.        # Let this line commented if you're using Puppet Dashboard
20.        #ensure => stopped,
21.        enable => false
22.      }
23.    }

```

A la ligne 9, nous utilisons une variable disponible dans les facts (côté client) pour qu'en fonction de la réponse, nous chargeons un fichier associé à à l'OS. Nous allons donc avoir un fichier de configuration accessible via Puppet sous la forme 'RedHat.puppet.conf'. Ensuite, le service, nous nous assurons qu'il soit bien stoppé au démarrage et qu'il est dans un état éteint pour le moment. En fait je ne souhaite pas que toutes les 30 minutes (valeur par défaut), il se déclenche et se synchronise, je trouve ça trop dangereux et préfère décider via d'autres mécanismes (SSH, Mcollective...) quant je souhaite qu'une synchronisation soit faite.

7.1.3 files

Dans files, nous allons avoir le fichier de configuration basique qui doit s'appliquer à toutes les machines de type RedHat :

```

/etc/puppet/modules/puppet/files/RedHat.puppet.conf

0. [main]
1. # The Puppet log directory.
2. # The default value is '$vardir/log'.
3. logdir = /var/log/puppet
4.
5. # Where Puppet PID files are kept.
6. # The default value is '$vardir/run'.
7. rundir = /var/run/puppet
8.
9. # Where SSL certificates are kept.
10. # The default value is '$confdir/ssl'.
11. ssl_dir = $vardir/ssl
12.
13. # Puppet master server
14. server = puppet-prd.deimos.fr
15.
16. # Add custom facts
17. pluginsync = true
18. plugin_source = puppet://$server/plugins
19. factpath = /var/lib/puppet/lib/facter
20.
21. [agent]
22. # The file in which puppetd stores a list of the classes
23. # associated with the retrieved configuration. Can be loaded in
24. # the separate ``puppet`` executable using the ``--loadclasses``
25. # option.
26. # The default value is '$confdir/classes.txt'.
27. classfile = $vardir/classes.txt
28.
29. # Where puppetd caches the local configuration. An
30. # extension indicating the cache format is added automatically.
31. # The default value is '$confdir/localconfig'.
32. localconfig = $vardir/localconfig
33.
34. # Reporting
35. report = true
36.
37. # Inspect reports for a compliance workflow
38. archive_files = true

```

7.2 resolvconf

J'ai fait ce module pour gérer le fichier de configuration resolv.conf. L'utilisation est assez simple, il va récupérer les informations des serveurs DNS renseignés dans le tableau disponible dans vars.pp du module base. Renseignez donc les serveurs DNS par défaut :

```

/etc/puppet/modules/base/manifests/vars.pp

0. # Default DNS servers
1. $dns_servers = [ '192.168.0.69', '192.168.0.27' ]

```

Vous pouvez surclasser ces valeurs directement au niveau d'un ou plusieurs noeuds si vous devez avoir des configuration spécifiques pour certains noeuds (dans le fichier servers.pp du module base) :

```

/etc/puppet/modules/base/manifests/servers.pp

0. # One server
1. node 'srv.deimos.fr' {
2.   $dns_servers = [ '127.0.0.1' ]
3.   include base::minimal
4. }

```

Créons l'arborescence :

```
mkdir
0. mkdir -p /etc/puppet/modules/resolvconf/{manifests,templates}
```

7.2.1 init.pp

```
/etc/puppet/modules/resolvconf/manifests/init.pp
0. /*
1. Resolv.conf Module for Puppet
2. Made by Pierre Mavro
3. */
4. class resolvconf {
5.     # Check OS and request the appropriate function
6.     case $::operatingsystem {
7.         'RedHat' : {
8.             include resolvconf::redhat
9.         }
10.        #'sunos': { include packages_defaults::solaris }
11.        default : {
12.            notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.        }
14.    }
15. }
```

7.2.2 redhat.pp

Voici la configuration pour Red Hat, j'utilise ici un fichier template, qui se renseignera avec les informations présente dans le tableau \$dns_servers :

```
/etc/puppet/modules/resolvconf/manifests/redhat.pp
0. /*
1. Resolvconf Module for Puppet
2. Made by Pierre Mavro
3. */
4. class resolvconf::redhat {
5.     # resolv.conf file
6.     file {
7.         "/etc/resolv.conf" :
8.             content => template("resolvconf/resolv.conf"),
9.             mode => 744,
10.            owner => root,
11.            group => root
12.        }
13.    }
```

7.2.3 templates

Et enfin mon fichier template resolv.conf :

```
/etc/puppet/modules/resolvconf/templates/resolv.conf
0. # Generated by Puppet
1. domain deimos.fr
2. search deimos.fr deimos.lan
3. <% dns_servers.each do |dnsval| ->
4. nameserver <%= dnsval %>
5. <% end ->
```

Ici nous avons une boucle ruby qui va parcourir le tableau \$dns_servers et qui va construire le fichier resolv.conf en insérant ligne par ligne 'nameserver' avec le serveur associé.

7.3 packages_defaults

J'utilise ce module pour qu'il installe ou désinstalle des packages dont j'ai absolument besoin sur toutes mes machines. Créons l'arborescence :

```
mkdir
0. mkdir -p /etc/puppet/modules/resolvconf/manifests
```

7.3.1 init.pp

```
/etc/puppet/modules/packages_defaults/manifests/init.pp
0. class packages_defaults {
1.     # Check OS and request the appropriate function
2.     case $::operatingsystem {
```


```

3.     'RedHat' : {
4.         include ::packages_defaults::redhat
5.     }
6.     #'sunos': { include packages_defaults::solaris }
7.     default : {
8.         notice("Module ${module_name} is not supported on ${::operatingsystem}")
9.     }
10. }
11. }

```

7.3.2 redhat.pp

J'aurais pu tout regrouper dans un seul bloc, mais par soucis de lisibilité sur les packages qui figurent dans la distribution et ceux que j'ai rajouté dans un repository custom, j'ai préféré faire une séparation :

 /etc/puppet/modules/packages_defaults/manifests/redhat.pp

```

0. # Red Hat Defaults packages
1. class packages_defaults::redhat
2. {
3.     # Set all default packages (embdeded in distribution) that need to be installed
4.     packages_install
5.     { [
6.         'nc',
7.         'tree',
8.         'telnet',
9.         'dialog',
10.        'freeipmi',
11.        'glibc-2.12-1.80.el6.i686'
12.    ] : }
13.     # Set all custom packages (not embdeded in distribution) that need to be installed
14.     packages_install
15.     { [
16.         'puppet',
17.         'tmux'
18.     ] : }
19. }
20. }

```


7.4 configurations_defaults

Ce module, tout comme le précédent est utilisé pour la configuration de l'OS livré en standard. Je souhaite en fait ici faire des ajustements sur des parties du système pure, sans vraiment rentrer sur un logiciel en particulier. Créons l'arborescence :

 mkdir

```
0. mkdir -p /etc/puppet/modules/configuration_defaults/{manifests,templates,lib/facter}
```

7.4.1 init.pp

 /etc/puppet/modules/configuration_defaults/manifests/init.pp

```


0. class configurations_defaults {
1.     import '*.pp'
2.
3.     # Configure common security parameters
4.     include configurations_defaults::common
5.
6.     # Check OS and request the appropriate function
7.     case $::operatingsystem {
8.         'RedHat' : {
9.             include configurations_defaults::redhat
10.        }
11.        default : {
12.            notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.        }
14.    }
15. }

```

Ici je viens charger tous les fichiers .pp au démarrage, puis appelle importe les configuration communes (common), ensuite j'applique les configurations spécifiques à chaque OS.

7.4.2 common.pp

Ici je souhaite avoir la même base de fichier motd pour toutes mes machines. Vous verrez par la suite pourquoi il figure en tant que template :

 /etc/puppet/modules/configuration_defaults/manifests/common.pp

```

0. class configurations_defaults::common
1. {
2.     # Motd banner for all servers
3.     file {
4.         '/etc/motd':
5.             ensure => present,
6.             content => template("configurations_defaults/motd"),
7.             mode => 644,
8.             owner => root,
9.             group => root
10.    }
11. }

```


7.4.3 redhat.pp

Je vais ici charger des options de sécurité, configurer automatiquement le bonding sur mes machines et une option de sysctl :

```
1. /etc/puppet/modules/configuration_defaults/manifests/redhat.pp
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
146.
147.
148.
149.
150.
151.
152.
153.
154.
155.
156.
157.
158.
159.
160.
161.
162.
163.
164.
165.
166.
167.
168.
169.
170.
171.
172.
173.
174.
175.
176.
177.
178.
179.
180.
181.
182.
183.
184.
185.
186.
187.
188.
189.
190.
191.
192.
193.
194.
195.
196.
197.
198.
199.
200.
201.
202.
203.
204.
205.
206.
207.
208.
209.
210.
211.
212.
213.
214.
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
229.
230.
231.
232.
233.
234.
235.
236.
237.
238.
239.
240.
241.
242.
243.
244.
245.
246.
247.
248.
249.
250.
251.
252.
253.
254.
255.
256.
257.
258.
259.
260.
261.
262.
263.
264.
265.
266.
267.
268.
269.
270.
271.
272.
273.
274.
275.
276.
277.
278.
279.
280.
281.
282.
283.
284.
285.
286.
287.
288.
289.
290.
291.
292.
293.
294.
295.
296.
297.
298.
299.
300.
301.
302.
303.
304.
305.
306.
307.
308.
309.
310.
311.
312.
313.
314.
315.
316.
317.
318.
319.
320.
321.
322.
323.
324.
325.
326.
327.
328.
329.
330.
331.
332.
333.
334.
335.
336.
337.
338.
339.
340.
341.
342.
343.
344.
345.
346.
347.
348.
349.
350.
351.
352.
353.
354.
355.
356.
357.
358.
359.
360.
361.
362.
363.
364.
365.
366.
367.
368.
369.
370.
371.
372.
373.
374.
375.
376.
377.
378.
379.
380.
381.
382.
383.
384.
385.
386.
387.
388.
389.
390.
391.
392.
393.
394.
395.
396.
397.
398.
399.
400.
401.
402.
403.
404.
405.
406.
407.
408.
409.
410.
411.
412.
413.
414.
415.
416.
417.
418.
419.
420.
421.
422.
423.
424.
425.
426.
427.
428.
429.
430.
431.
432.
433.
434.
435.
436.
437.
438.
439.
440.
441.
442.
443.
444.
445.
446.
447.
448.
449.
450.
451.
452.
453.
454.
455.
456.
457.
458.
459.
460.
461.
462.
463.
464.
465.
466.
467.
468.
469.
470.
471.
472.
473.
474.
475.
476.
477.
478.
479.
480.
481.
482.
483.
484.
485.
486.
487.
488.
489.
490.
491.
492.
493.
494.
495.
496.
497.
498.
499.
500.
501.
502.
503.
504.
505.
506.
507.
508.
509.
510.
511.
512.
513.
514.
515.
516.
517.
518.
519.
520.
521.
522.
523.
524.
525.
526.
527.
528.
529.
530.
531.
532.
533.
534.
535.
536.
537.
538.
539.
540.
541.
542.
543.
544.
545.
546.
547.
548.
549.
550.
551.
552.
553.
554.
555.
556.
557.
558.
559.
560.
561.
562.
563.
564.
565.
566.
567.
568.
569.
570.
571.
572.
573.
574.
575.
576.
577.
578.
579.
580.
581.
582.
583.
584.
585.
586.
587.
588.
589.
590.
591.
592.
593.
594.
595.
596.
597.
598.
599.
600.
601.
602.
603.
604.
605.
606.
607.
608.
609.
610.
611.
612.
613.
614.
615.
616.
617.
618.
619.
620.
621.
622.
623.
624.
625.
626.
627.
628.
629.
630.
631.
632.
633.
634.
635.
636.
637.
638.
639.
640.
641.
642.
643.
644.
645.
646.
647.
648.
649.
650.
651.
652.
653.
654.
655.
656.
657.
658.
659.
660.
661.
662.
663.
664.
665.
666.
667.
668.
669.
670.
671.
672.
673.
674.
675.
676.
677.
678.
679.
680.
681.
682.
683.
684.
685.
686.
687.
688.
689.
690.
691.
692.
693.
694.
695.
696.
697.
698.
699.
700.
701.
702.
703.
704.
705.
706.
707.
708.
709.
710.
711.
712.
713.
714.
715.
716.
717.
718.
719.
720.
721.
722.
723.
724.
725.
726.
727.
728.
729.
730.
731.
732.
733.
734.
735.
736.
737.
738.
739.
740.
741.
742.
743.
744.
745.
746.
747.
748.
749.
750.
751.
752.
753.
754.
755.
756.
757.
758.
759.
760.
761.
762.
763.
764.
765.
766.
767.
768.
769.
770.
771.
772.
773.
774.
775.
776.
777.
778.
779.
780.
781.
782.
783.
784.
785.
786.
787.
788.
789.
790.
791.
792.
793.
794.
795.
796.
797.
798.
799.
800.
801.
802.
803.
804.
805.
806.
807.
808.
809.
810.
811.
812.
813.
814.
815.
816.
817.
818.
819.
820.
821.
822.
823.
824.
825.
826.
827.
828.
829.
830.
831.
832.
833.
834.
835.
836.
837.
838.
839.
840.
841.
842.
843.
844.
845.
846.
847.
848.
849.
850.
851.
852.
853.
854.
855.
856.
857.
858.
859.
860.
861.
862.
863.
864.
865.
866.
867.
868.
869.
870.
871.
872.
873.
874.
875.
876.
877.
878.
879.
880.
881.
882.
883.
884.
885.
886.
887.
888.
889.
890.
891.
892.
893.
894.
895.
896.
897.
898.
899.
900.
901.
902.
903.
904.
905.
906.
907.
908.
909.
910.
911.
912.
913.
914.
915.
916.
917.
918.
919.
920.
921.
922.
923.
924.
925.
926.
927.
928.
929.
930.
931.
932.
933.
934.
935.
936.
937.
938.
939.
940.
941.
942.
943.
944.
945.
946.
947.
948.
949.
950.
951.
952.
953.
954.
955.
956.
957.
958.
959.
960.
961.
962.
963.
964.
965.
966.
967.
968.
969.
970.
971.
972.
973.
974.
975.
976.
977.
978.
979.
980.
981.
982.
983.
984.
985.
986.
987.
988.
989.
990.
991.
992.
993.
994.
995.
996.
997.
998.
999.
1000.
```

7.4.4 security.pp

Vous allez voir que ce fichier fait pas mal de choses :

```
1. /etc/puppet/modules/configuration_defaults/manifests/security.pp
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
146.
147.
148.
149.
150.
151.
152.
153.
154.
155.
156.
157.
158.
159.
160.
161.
162.
163.
164.
165.
166.
167.
168.
169.
170.
171.
172.
173.
174.
175.
176.
177.
178.
179.
180.
181.
182.
183.
184.
185.
186.
187.
188.
189.
190.
191.
192.
193.
194.
195.
196.
197.
198.
199.
200.
201.
202.
203.
204.
205.
206.
207.
208.
209.
210.
211.
212.
213.
214.
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
229.
230.
231.
232.
233.
234.
235.
236.
237.
238.
239.
240.
241.
242.
243.
244.
245.
246.
247.
248.
249.
250.
251.
252.
253.
254.
255.
256.
257.
258.
259.
260.
261.
262.
263.
264.
265.
266.
267.
268.
269.
270.
271.
272.
273.
274.
275.
276.
277.
278.
279.
280.
281.
282.
283.
284.
285.
286.
287.
288.
289.
290.
291.
292.
293.
294.
295.
296.
297.
298.
299.
300.
301.
302.
303.
304.
305.
306.
307.
308.
309.
310.
311.
312.
313.
314.
315.
316.
317.
318.
319.
320.
321.
322.
323.
324.
325.
326.
327.
328.
329.
330.
331.
332.
333.
334.
335.
336.
337.
338.
339.
340.
341.
342.
343.
344.
345.
346.
347.
348.
349.
350.
351.
352.
353.
354.
355.
356.
357.
358.
359.
360.
361.
362.
363.
364.
365.
366.
367.
368.
369.
370.
371.
372.
373.
374.
375.
376.
377.
378.
379.
380.
381.
382.
383.
384.
385.
386.
387.
388.
389.
390.
391.
392.
393.
394.
395.
396.
397.
398.
399.
400.
401.
402.
403.
404.
405.
406.
407.
408.
409.
410.
411.
412.
413.
414.
415.
416.
417.
418.
419.
420.
421.
422.
423.
424.
425.
426.
427.
428.
429.
430.
431.
432.
433.
434.
435.
436.
437.
438.
439.
440.
441.
442.
443.
444.
445.
446.
447.
448.
449.
450.
451.
452.
453.
454.
455.
456.
457.
458.
459.
460.
461.
462.
463.
464.
465.
466.
467.
468.
469.
470.
471.
472.
473.
474.
475.
476.
477.
478.
479.
480.
481.
482.
483.
484.
485.
486.
487.
488.
489.
490.
491.
492.
493.
494.
495.
496.
497.
498.
499.
500.
501.
502.
503.
504.
505.
506.
507.
508.
509.
510.
511.
512.
513.
514.
515.
516.
517.
518.
519.
520.
521.
522.
523.
524.
525.
526.
527.
528.
529.
530.
531.
532.
533.
534.
535.
536.
537.
538.
539.
540.
541.
542.
543.
544.
545.
546.
547.
548.
549.
550.
551.
552.
553.
554.
555.
556.
557.
558.
559.
560.
561.
562.
563.
564.
565.
566.
567.
568.
569.
570.
571.
572.
573.
574.
575.
576.
577.
578.
579.
580.
581.
582.
583.
584.
585.
586.
587.
588.
589.
590.
591.
592.
593.
594.
595.
596.
597.
598.
599.
600.
601.
602.
603.
604.
605.
606.
607.
608.
609.
610.
611.
612.
613.
614.
615.
616.
617.
618.
619.
620.
621.
622.
623.
624.
625.
626.
627.
628.
629.
630.
631.
632.
633.
634.
635.
636.
637.
638.
639.
640.
641.
642.
643.
644.
645.
646.
647.
648.
649.
650.
651.
652.
653.
654.
655.
656.
657.
658.
659.
660.
661.
662.
663.
664.
665.
666.
667.
668.
669.
670.
671.
672.
673.
674.
675.
676.
677.
678.
679.
680.
681.
682.
683.
684.
685.
686.
687.
688.
689.
690.
691.
692.
693.
694.
695.
696.
697.
698.
699.
700.
701.
702.
703.
704.
705.
706.
707.
708.
709.
710.
711.
712.
713.
714.
715.
716.
717.
718.
719.
720.
721.
722.
723.
724.
725.
726.
727.
728.
729.
730.
731.
732.
733.
734.
735.
736.
737.
738.
739.
740.
741.
742.
743.
744.
745.
746.
747.
748.
749.
750.
751.
752.
753.
754.
755.
756.
757.
758.
759.
760.
761.
762.
763.
764.
765.
766.
767.
768.
769.
770.
771.
772.
773.
774.
775.
776.
777.
778.
779.
780.
781.
782.
783.
784.
785.
786.
787.
788.
789.
790.
791.
792.
793.
794.
795.
796.
797.
798.
799.
800.
801.
802.
803.
804.
805.
806.
807.
808.
809.
810.
811.
812.
813.
814.
815.
816.
817.
818.
819.
820.
821.
822.
823.
824.
825.
826.
827.
828.
829.
830.
831.
832.
833.
834.
835.
836.
837.
838.
839.
840.
841.
842.
843.
844.
845.
846.
847.
848.
849.
850.
851.
852.
853.
854.
855.
856.
857.
858.
859.
860.
861.
862.
863.
864.
865.
866.
867.
868.
869.
870.
871.
872.
873.
874.
875.
876.
877.
878.
879.
880.
881.
882.
883.
884.
885.
886.
887.
888.
889.
890.
891.
892.
893.
894.
895.
896.
897.
898.
899.
900.
901.
902.
903.
904.
905.
906.
907.
908.
909.
910.
911.
912.
913.
914.
915.
916.
917.
918.
919.
920.
921.
922.
923.
924.
925.
926.
927.
928.
929.
930.
931.
932.
933.
934.
935.
936.
937.
938.
939.
940.
941.
942.
943.
944.
945.
946.
947.
948.
949.
950.
951.
952.
953.
954.
955.
956.
957.
958.
959.
960.
961.
962.
963.
964.
965.
966.
967.
968.
969.
970.
971.
972.
973.
974.
975.
976.
977.
978.
979.
980.
981.
982.
983.
984.
985.
986.
987.
988.
989.
990.
991.
992.
993.
994.
995.
996.
997.
998.
999.
1000.
```

Quelques explications s'imposent :

- Manage Root passwords : On définit le mot de passe root souhaité sous forme md5 et sha1. En fonction de ce qu'il y aura de configuré sur la machine, il configurera le mot de passe souhaité. Pour cette détection j'utilise un facter (passwd_algorithm.rb)
- Enable auditd service : on s'assure que le service auditd démarrera bien au boot et tourne actuellement
- Comment unwanted sysctl lines : nous demandons que certaines lignes présentes dans sysctl soient commentées si elles existent
- Add security sysctl values : nous ajoutons des règles sysctl, ainsi que leur assignons une valeur
- Deny kernel read to others users : j'ai créer ici un facter, qui permet de vérifier les droits des fichiers kernel (kernel_rights.rb)
- Change opened file descriptor value : permet d'utiliser la fonction limits_conf pour gérer le fichier limits.conf. Ici j'ai changé la valeur par défaut des file descriptor et on rajoute une petite sécurité pour éviter les fork bomb.

7.4.5 facter

Nous allons insérer ici les facters qui vont nous servir pour certaines fonctions demandées ci dessus.

7.4.5.1 passwd_algorithm.rb

Ce facter va déterminer l'algorithme utilisé pour l'authentification :

```
etc/puppet/modules/configuration_defaults/lib/facter/passwd_algorithm.rb

0. # Get Passwd Algorithm
1. Facter.add(:passwd_algorithm) do
2.   setcode do
3.     Facter::Util::Resolution.exec("grep ^PASSWDALGORITHM /etc/sysconfig/authconfig | awk -F=' ' { print $2 }")
4.   end
5. end
```

7.4.5.2 kernel_rights.rb

Ce facter va déterminer si n'importe quel utilisateur a le droit de lire les kernels installés sur la machine courante :

```
etc/puppet/modules/configuration_defaults/lib/facter/kernel_rights.rb

0. # Get security rights
1. Facter.add(:kernel_security_rights) do
2.   # Get kernel files where rights will be checked
3.   kernel_files = Dir.glob("/boot/(vmlinuz,System.map)*")
4.   current_rights=1
5.
6.   # Check each files
7.   kernel_files.each do |file|
8.     # Get file mode
9.     full_rights = sprintf("%o", File.stat(file).mode)
10.    # Get last number (corresponding to other rights)
11.    other_rights = Integer(full_rights) % 10
12.    # Check if other got read rights
13.    if other_rights >= 4
14.      current_rights=0
15.    end
16.  end
17.  setcode do
18.    # Set kernel_security_rights to 1 if read value is detected
19.    current_rights
20.  end
21. end
```

7.4.5.3 get_network_infos.rb

Ce facter permet de récupérer l'ip courante sur eth0, le netmask et la gateway :

```
etc/puppet/modules/configuration_defaults/lib/facter/get_network_infos.rb

0. # Get public IP address
1. Facter.add(:public_ip) do
2.   setcode do
3.     # Get bond0 ip if exist
4.     if File.exist? "/proc/sys/net/ipv4/conf/bond0"
5.       Facter::Util::Resolution.exec("ip addr show dev bond0 | awk '/inet/{print $2}' | head -1 | sed 's/\\/\\.*/'")
6.     else
7.       # Or eth0 ip if exist
8.       if File.exist? "/proc/sys/net/ipv4/conf/eth0"
9.         Facter::Util::Resolution.exec("ip addr show dev eth0 | awk '/inet/{print $2}' | head -1 | sed 's/\\/\\.*/'")
10.      else
11.        # Else return error
12.        'unknown (puppet issue)'
13.      end
14.    end
15.  end
16. end
17.
18. # Get netmask on the fist interface
19. Facter.add(:public_netmask) do
20.   setcode do
21.     # Get bond0 netmask if exist
22.     if File.exist? "/proc/sys/net/ipv4/conf/bond0"
23.       Facter::Util::Resolution.exec("ifconfig bond0 | awk '/inet/{print $4}' | sed 's/\\.*/'")
24.     else
25.       # Or eth0 netmask if exist
26.       if File.exist? "/proc/sys/net/ipv4/conf/eth0"
27.         Facter::Util::Resolution.exec("ifconfig eth0 | awk '/inet/{print $4}' | sed 's/\\.*/'")
28.       else
29.         # Else set a default netmask
30.         '255.255.255.0'
31.       end
32.     end
33.   end
34. end
35.
36. # Get default gateway
37. Facter.add(:default_gateway) do
38.   setcode do
39.     Facter::Util::Resolution.exec("ip route | awk '/default/{print $3}'")
40.   end
41. end
```

7.4.6 network.pp

Ici nous allons charger la configuration du bonding et d'autres choses liées au réseau :

```
etc/puppet/modules/configuration_defaults/manifests/network.pp
```

```


0. class configurations_defaults::redhat::network inherits configurations_defaults::redhat
1. {
2.   # Disable network interface renaming
3.   Augeas {
4.     "grub_udev_net" :
5.       context => "/files/etc/grub.conf",
6.       changes => "set title[1]/kernel/biosdevname 0"
7.     }
8.
9.   # Load bonding module at boot
10.  line {
11.    'load_bonding':
12.      file => '/etc/modprobe.d/bonding.conf',
13.      line => 'alias bond0 bonding',
14.      ensure => present
15.    }
16.
17.  # Bonded master interface - static
18.  network::bond::static {
19.    "bond0" :
20.      ipaddress => "$::public_ip",
21.      netmask => "$::public_netmask",
22.      gateway => "$::default_gateway",
23.      bonding_opts => "mode=active-backup",
24.      ensure => "up"
25.    }
26.
27.  # Bonded slave interface - static
28.  network::bond::slave {
29.    "eth0" :
30.      macaddress => $::macaddress_eth0,
31.      master => "bond0",
32.    }
33.
34.  # Bonded slave interface - static
35.  network::bond::slave {
36.    "eth1" :
37.      macaddress => $::macaddress_eth1,
38.      master => "bond0",
39.    }
40. }

```

- Disable network interface renaming : nous ajoutons un argument et mettons sa valeur à 0 dans grub pour qu'il ne renomme pas les interfaces et les laisse en ethX. J'ai fait un article qui parle de ça si ça vous intéresse.
- Load bonding module at boot : On s'assure que le module bonding sera bien chargé au boot de la machine et qu'un alias sur bond0 existe
- Bonded interfaces : Je vous renvoie sur le module bonding disponible sur Puppet Forge (<http://forge.puppetlabs.com/razorsedge/network>) , ainsi qu'à la documentation sur le bonding si vous savez pas ce que c'est. J'ai également créé un facter (`get_network_infos.rb`) pour cela afin de récupérer l'interface publique (eth0, sur laquelle je me connecterais), le netmask et gateway déjà présent et configuré sur la machine

7.4.7 templates

Nous en avons parlé plus haut, nous gérons une template pour motd afin d'afficher en plus d'un texte, le hostname de la machine sur laquelle vous vous connectez (ligne 16) :

 /etc/puppet/modules/configuration_defaults/templates/motd

```

0. =====
1. This is an official computer system and is the property of Deimos. It
2. is for authorized users only. Unauthorized users are prohibited. Users
3. (authorized or unauthorized) have no explicit or implicit expectation of
4. privacy. Any or all uses of this system may be subject to one or more of
5. the following actions: interception, monitoring, recording, auditing,
6. inspection and disclosing to security personnel and law enforcement
7. personnel, as well as authorized officials of other agencies, both domestic
8. and foreign. By using this system, the user consents to these actions.
9. Unauthorized or improper use of this system may result in administrative
10. disciplinary action and civil and criminal penalties. By accessing this
11. system you indicate your awareness of and consent to these terms and
12. conditions of use. Discontinue access immediately if you do not agree to
13. the conditions stated in this notice.
14. =====
15.
16. <%= hostname %>

```

7.5 OpenSSH - 1

Voici un premier exemple pour OpenSSH. Créons l'arborescence :

 mkdir

```
0. mkdir -p /etc/puppet/modules/openssh/{manifests,templates,lib/facter}
```

7.5.1 init.pp

 /etc/puppet/modules/openssh/manifests/init.pp

```

0. /*
1. OpenSSH Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openssh {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'RedHat' : {
8.       include openssh::redhat
9.     }
10.    #'sunos': { include packages_defaults::solaris }
11.    default : {
12.      notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.    }
14.  }

```

```
15. }
```

7.5.2 redhat.pp

Nous chargeons ici tout ce dont nous avons besoin, puis chargeons le common car il faut qu'OpenSSH soit installé et configuré avant de passer à la partie commune :

```
./etc/puppet/modules/openssh/manifests/redhat.pp

0. /*
1. OpenSSH Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openssh::redhat {
5.     # Install ssh package
6.     package {
7.         'openssh-server' :
8.             ensure => present
9.     }
10.
11.     # SSHd config file
12.     file {
13.         "/etc/ssh/sshd_config" :
14.             source => "puppet:///modules/openssh/sshd_config.${::operatingsystem}",
15.             mode => 600,
16.             owner => root,
17.             group => root,
18.             notify => Service["sshd"]
19.     }
20.
21.     service {
22.         'sshd' :
23.             enable => true,
24.             ensure => running,
25.             require => File['/etc/ssh/sshd_config']
26.     }
27.
28.     include openssh::common
29. }
```

Dans la partie service, il y a une information importante (require), qui redémarrera le service si le fichier de configuration change.

7.5.3 common.pp

Ici, nous nous assurons que le dossier ou stocker des clefs SSH est bien présent avec les bons droits, puis nous faisons appel à un autre fichier qui va contenir toutes les clefs que nous souhaitons exporter :

```
./etc/puppet/modules/openssh/manifests/common.pp

0. /*
1. OpenSSH Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openssh::common {
5.     # Check that .ssh directory exist with correct rights
6.     file {
7.         "${::home_root}/.ssh" :
8.             ensure => directory,
9.             mode => 0700,
10.            owner => root,
11.            group => root
12.        }
13.
14.        # Load all public keys
15.        include openssh::ssh_keys
16.    }
```

La directive 'home_root' est généré depuis un facteur fournis ci dessous.

7.5.4 factor

Voici le facteur qui permet de récupérer le home du user root :

```
./etc/puppet/modules/openssh/lib/factor/home_root.rb

0. # Get Home directory
1. Factor.add("home_root") do
2.     setcode do
3.         Factor::Util::Resolution.exec("echo ~root")
4.     end
5. end
```

7.5.5 ssh_keys.pp

Ici je rajoute les clefs, que ce soit pour l'accès depuis d'autres serveurs ou bien des utilisateurs :

```
./etc/puppet/modules/openssh/manifests/ssh_keys.pp

0. /*
1. OpenSSH Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openssh::ssh_keys inherits openssh::common {
5.     #####
```


```

6. # Servers
7. #####
8.
9. # Puppet master
10. ssh_add_key {
11.   'puppet_root' :
12.     user => 'root',
13.     key => 'AAAA...'
14. }
15.
16. #####
17. # Sys-Admins
18. #####
19.
20. # Pierre Mavro
21. ssh_add_key {
22.   'pmavro_root' :
23.     user => 'root',
24.     key => 'AAAA...'
25. }
26. }

```

7.5.6 files

Le fichier de configuration OpenSSH pour Red Hat :

 /etc/puppet/modules/openssh/files/sshd_config.RedHat

```

0. # $OpenBSD: sshd_config,v 1.80 2008/07/02 02:24:18 djm Exp $
1.
2. # This is the sshd server system-wide configuration file. See
3. # sshd_config(5) for more information.
4.
5. # This sshd was compiled with PATH=/usr/local/bin:/bin:/usr/bin
6.
7. # The strategy used for options in the default sshd_config shipped with
8. # OpenSSH is to specify options with their default value where
9. # possible, but leave them commented. Uncommented options change a
10. # default value.
11.
12. Port 22
13. #AddressFamily any
14. #ListenAddress 0.0.0.0
15. #ListenAddress ::
16.
17. # Disable legacy (protocol version 1) support in the server for new
18. # installations. In future the default will change to require explicit
19. # activation of protocol 1
20. Protocol 2
21.
22. # HostKey for protocol version 1
23. #HostKey /etc/ssh/ssh_host_key
24. # HostKeys for protocol version 2
25. #HostKey /etc/ssh/ssh_host_rsa_key
26. #HostKey /etc/ssh/ssh_host_dsa_key
27.
28. # Lifetime and size of ephemeral version 1 server key
29. #KeyRegenerationInterval 1h
30. #ServerKeyBits 1024
31.
32. # Logging
33. # obsoletes QuietMode and FascistLogging
34. #SyslogFacility AUTH
35. #SyslogFacility AUTHPRIV
36. #LogLevel INFO
37.
38. # Authentication:
39.
40. LoginGraceTime 2m
41. PermitRootLogin without-password
42. #StrictModes yes
43. #MaxAuthTries 6
44. #MaxSessions 10
45.
46. #RSAAuthentication yes
47. #PubkeyAuthentication yes
48. #AuthorizedKeysFile .ssh/authorized_keys
49. #AuthorizedKeysCommand none
50. #AuthorizedKeysCommandRunAs nobody
51.
52. # For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
53. #RhostsRSAAuthentication no
54. # similar for protocol version 2
55. #HostbasedAuthentication no
56. # Change to yes if you don't trust ~/.ssh/known_hosts for
57. # RhostsRSAAuthentication and HostbasedAuthentication
58. #IgnoreUserKnownHosts no
59. # Don't read the user's ~/.rhosts and ~/.shosts files
60. #IgnoreRhosts yes
61.
62. # To disable tunneled clear text passwords, change to no here!
63. #PasswordAuthentication yes
64. #PermitEmptyPasswords no
65. PasswordAuthentication yes
66.
67. # Change to no to disable s/key passwords
68. #ChallengeResponseAuthentication yes
69. ChallengeResponseAuthentication no
70.
71. # Kerberos options
72. #KerberosAuthentication no
73. #KerberosOrLocalPasswd yes
74. #KerberosTicketCleanup yes
75. #KerberosGetAFSToken no
76. #KerberosUseUserok yes
77.
78. # GSSAPI options
79. # Disable GSSAPI to avoid login slowdown
80. GSSAPIAuthentication no
81. #GSSAPIAuthentication yes
82. #GSSAPICleanupCredentials yes
83. #GSSAPICleanupCredentials no
84. #GSSAPIStrictAcceptorCheck yes
85. #GSSAPIKeyExchange no
86.
87. # Set this to 'yes' to enable PAM authentication, account processing,
88. # and session processing. If this is enabled, PAM authentication will
89. # be allowed through the ChallengeResponseAuthentication and
90. # PasswordAuthentication. Depending on your PAM configuration,
91. # PAM authentication via ChallengeResponseAuthentication may bypass
92. # the setting of "PermitRootLogin without-password".
93. # If you just want the PAM account and session checks to run without
94. # PAM authentication, then enable this but set PasswordAuthentication
95. # and ChallengeResponseAuthentication to 'no'.

```

```

96. #UsePAM no
97. UsePAM yes
98.
99. # Accept locale-related environment variables
100. AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
101. AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
102. AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
103. AcceptEnv XMODIFIERS
104.
105. # For security reasons, deny tcp forwarding
106. AllowTcpForwarding no
107. X11Forwarding no
108. # Disable DNS usage to avoid login slowdown
109. UseDNS no
110. # Disconnect client if they are idle
111. ClientAliveInterval 600
112. ClientAliveCountMax 0
113.
114. #AllowAgentForwarding yes
115. #GatewayPorts no
116. #X11Forwarding no
117. #X11DisplayOffset 10
118. #X11UseLocalhost yes
119. #PrintMotd yes
120. #PrintLastLog yes
121. #TCPKeepAlive yes
122. #UseLogin no
123. #UsePrivilegeSeparation yes
124. #PermitUserEnvironment no
125. #Compression delayed
126. #ShowPatchLevel no
127. #PidFile /var/run/sshd.pid
128. #MaxStartups 10
129. #PermitTunnel no
130. #ChrootDirectory none
131.
132. # no default banner path
133. #Banner none
134.
135. # override default of no subsystems
136. Subsystem sftp /usr/libexec/openssh/sftp-server
137.
138. # Example of overriding settings on a per-user basis
139. #Match User anoncvs
140. # X11Forwarding no
141. # AllowTcpForwarding no
142. # ForceCommand cvs server


```

7.6 OpenSSH - 2

Voici un deuxième exemple pour OpenSSH, un peu différent. Vous devez initialiser le module avant de continuer.

7.6.1 init.pp

Ici je veux que ce soit mon fichier sshd_config qui m'intéresse :

 /etc/puppet/modules/ssh/manifests/init.pp

```

0. #ssh.pp
1.
2. # SSH Class with all includes
3. class ssh {
4.   $ssh_default_port = ["22"]
5.   $ssh_allowed_users = "root"
6.   include ssh::config, ssh::key, ssh::service
7. }
8.
9. # SSHD Config file
10. class ssh::config {
11.   file {
12.     name => $operatingsystem ? {
13.       Solaris => "/etc/ssh/sshd_config",
14.       default => "/etc/ssh/sshd_config"
15.     },
16.   }
17.
18.   # Using templates for sshd_config
19.   file { sshd_config:
20.     content => $operatingsystem ? {
21.       default => template("ssh/sshd_config"),
22.       Solaris => template("ssh/sshd_config.solaris"),
23.     }
24.   }
25. }
26.
27. # SSH Key exchange
28. class ssh::key {
29.   $basedir = $operatingsystem ? {
30.     Solaris => "/.ssh",
31.     Debian => "/root/.ssh",
32.     Redhat => "/root/.ssh",
33.   }
34.
35.   # Make sur .ssh exist in root home dir
36.   file { "$basedir/":
37.     ensure => directory,
38.     mode => 0700,
39.     owner => root,
40.     group => root,
41.     ignore => '.svn'
42.   }
43.
44.   # Check if authorized_keys key file exist or create empty one
45.   file { "$basedir/authorized_keys":
46.     ensure => present,
47.   }
48.
49.   # Check this line exist
50.   line { ssh_key:
51.     file => "$basedir/authorized_keys",
52.     line => "ssh-dss AAAAB3NzaC1...xG3ZA== root@puppet",
53.     ensure => present,
54.   }
55. }
56.
57. # Check servoce status
58. class ssh::service {
59.   service { ssh:
60.     name => $operatingsystem ? {
61.       Solaris => "svc:/network/ssh:default",
62.       default => ssh
63.     },
64.     ensure => running,
65.     enable => true

```

```
66. }
67. }
```

Ensuite, par rapport à sudo, j'ai un notify qui permet d'automatiquement redémarrer le service lorsque le fichier est remplacé par une nouvelle version. C'est le service ssh avec l'option "ensure => running", qui va permettre la détection du changement de version et redémarrage.

7.6.2 templates

Du fait que nous utilisons des templates, nous allons avoir besoin de créer un dossier templates :

```
mkdir
```

```
mkdir -p /etc/puppet/modules/ssh/templates
```

Ensuite nous allons créer 2 fichiers (sshd_config et sshd_config.solaris) car les configuration ne se comportent pas de la même manière (OpenSSH vs Sun SSH. Je n'aborderais cependant ici que la partie OpenSSH :

```
/etc/puppet/modules/ssh/templates/sshd_config
```

```
## Package generated configuration file
## See the sshd(8) manpage for details
#
# What ports, IPs and protocols we listen for
<# ssh_default_port.each do |val| ->
Port <%= val %>
<# end ->
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes
# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768
# Logging
SyslogFacility AUTH
LogLevel INFO
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
#
#RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile  %h/.ssh/authorized_keys
#
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
#hostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes
# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no
#
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no
#
# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes
#
# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosLocalPasswd yes
#KerberosTicketCleanup yes
#
# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no
#MaxStartups 10:30:60
#Banner /etc/issue.net
# Allow client to pass locale environment variables
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
# AllowUsers <%= ssh_allowed_users %>
```

7.7 SELinux

Si vous souhaitez en connaître plus sur SELinux, je vous invite à regarder cette documentation. Créons l'arborescence :

```
mkdir
```

```
0. mkdir -p /etc/puppet/modules/selinux/manifests
```

7.7.1 init.pp

```
/etc/puppet/modules/selinux/manifests/init.pp
```


```

0. class selinux {
1.   # Check OS and request the appropriate function
2.   case $::operatingsystem {
3.     'RedHat' : {
4.       include selinux::redhat
5.     }
6.   }
7. }

```

7.7.2 redhat.pp

Ici, je vais utiliser une fonction (augeas), qui va me permettre de passer la valeur de la variable 'SELINUX' à 'disabled', car je souhaite désactiver ce module :

 /etc/puppet/modules/selinux/manifests/redhat.pp

```

0. class selinux::redhat
1. {
2.   # Disable SELinux
3.   augeas {
4.     "selinux" :
5.     context => "/files/etc/sysconfig/selinux/",
6.     changes => "set SELINUX disabled"
7.   }
8. }

```


7.8 Grub

Il existe beaucoup d'options de Grub et son utilisation peut varier d'un système à l'autre. Je vous recommande cette documentation avant d'attaquer ceci. Créons l'arborescence :

 mkdir

```
0. mkdir -p /etc/puppet/modules/grub/manifests
```

7.8.1 init.pp

 /etc/puppet/modules/grub/manifests/init.pp

```

0. /*
1. Grub Module for Puppet
2. Made by Pierre Mavro
3. */
4. class grub {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'RedHat' : {
8.       include ::grub::redhat
9.     }
10.    #'sunos': { include packages_defaults::solaris }
11.    default : {
12.      notice("Module ${module_name} is not supported on ${{:operatingsystem}")
13.    }
14.  }
15. }

```

7.8.2 redhat.pp

Les choses vont se compliquer un peu plus ici. J'utilise encore le module Augeas pour supprimer les arguments quiet et rhgb des kernels disponibles dans grub.conf :

 /etc/puppet/modules/grub/manifests/redhat.pp

```

0. /*
1. Grub Module for Puppet
2. Made by Pierre Mavro
3. */
4. class grub::redhat
5. {
6.   # Remove unwanted parameters parameter
7.   augeas {
8.     "grub" :
9.     context => "/files/etc/grub.conf",
10.    changes => [
11.      "remove title[*]/kernel/quiet",
12.      "remove title[*]/kernel/rhgb"
13.    ]
14.  }
15. }

```


7.9 Kdump

Créons l'arborescence :

 mkdir


```
0. mkdir -p /etc/puppet/modules/kdump/manifests
```


7.9.1 init.pp

 /etc/puppet/modules/kdump/manifests/init.pp

```
0. /*
1. Kdump Module for Puppet
2. Made by Pierre Mavro
3. */
4. class kdump {
5.     # Check OS and request the appropriate function
6.     case $::operatingsystem {
7.         'RedHat' : {
8.             include kdump::redhat
9.         }
10.         default : {
11.             notice("Module ${module_name} is not supported on ${::operatingsystem}")
12.         }
13.     }
14. }
```

7.9.2 redhat.pp

Voici la configuration que je souhaite :

 /etc/puppet/modules/kdump/manifests/redhat.pp

```
0. /*
1. Kdump Module for Puppet
2. Made by Pierre Mavro
3. */
4. class kdump::redhat
5. {
6.     # Install kexec-tools
7.     package { 'kexec-tools':
8.         ensure => 'installed'
9.     }
10.
11.     # Be sure that service is set to start at boot
12.     service {
13.         'kdump':
14.             enable => true
15.     }
16.
17.     # Set crashkernel in grub.conf to the good size (not auto)
18.     Augeas {
19.         "grub_kdump" :
20.             context => '/files/etc/grub.conf',
21.             changes => [
22.                 "set title[1]/kernel/crashkernel 128M"
23.             ]
24.     }
25.
26.     # Set location of crash dumps
27.     line {
28.         'var_crash':
29.             file => '/etc/kdump.conf',
30.             line => 'path \var\crash',
31.             ensure => uncomment
32.     }
33. }
```

- Install kexec-tools : je m'assure que le package est bien installé
- Be sure that service is set to start at boot : je m'assure qu'il est bien activé au boot. Pour information, je ne vérifie pas qu'il tourne puisque cela nécessite un redémarrage s'il n'était pas présent.
- Set crashkernel in grub.conf to the good size (not auto) : Met la valeur 128M à l'argument crashkernel du premier kernel trouvé dans le fichier grub.conf. Il n'est pas possible de spécifier '8' comme pour un remove dans Augeas. Cependant, vu qu'à chaque mise à jour du kernel, tous les autres en hériteront, ce n'est pas un souci :-)
- Set location of crash dumps : nous spécifions dans la configuration de kdump qu'une ligne est bien décommentée et qu'elle a le path souhaité pour les crash dump.

7.10 Tools

Ceci n'est pas un logiciel, mais plutôt un module qui me sert à envoyer tous mes scripts d'admin, mes outils etc... Créons l'arborescence :

 mkdir

```
0. mkdir -p /etc/puppet/modules/tools/{manifests,files}
```

7.10.1 init.pp

 /etc/puppet/modules/tools/manifests/init.pp

```
0. class tools {
1.     # Check OS and request the appropriate function
2.     case $::operatingsystem {
3.         'RedHat' : {
4.             include ::tools::redhat
5.         }
6.         #Sunos : { include packages_defaults::solaris }
7.         default : {
```

```

8.     notice("Module ${module_name} is not supported on ${::operatingsystem}")
9.   }
10. }
11. }

```

7.10.2 redhat.pp

Nous allons voir ici différentes manière d'ajouter des fichiers :

```

/etc/puppet/modules/tools/manifests/redhat.pp

0. class tools::redhat {
1.   # Check that scripts folder exist
2.   file {
3.     "/etc/scripts" :
4.       ensure => directory,
5.       mode => 0755,
6.       owner => root,
7.       group => root
8.     }
9.
10.    # Synchro admin-scripts
11.    file {
12.      "/etc/scripts/admin-scripts" :
13.        ensure => directory,
14.        mode => 0755,
15.        owner => root,
16.        group => root,
17.        source => "puppet:///modules/tools/admin-scripts/",
18.        purge => true,
19.        force => true,
20.        recurse => true,
21.        ignore => ".svn",
22.        backup => false
23.      }
24.
25.      # Fast reboot command
26.      file {
27.        "/usr/bin/fastreboot" :
28.          source => "puppet:///modules/tools/fastreboot",
29.          mode => 744,
30.          owner => root,
31.          group => root
32.        }
33.      }

```

- Check that scripts folder exist : Je m'assure que mon dossier existe avec les bons droits, avant d'y déposer des fichiers.
- Synchro admin-scripts : Je viens copier un répertoire complet avec son contenu :
 - purge => true : Je m'assure que tout ce qui n'est pas dans mon puppet files, doit disparaître côté serveur. Donc si vous avez manuellement ajouté un fichier dans le dossier /etc/scripts/admin-scripts, il disparaîtra.
 - force => true : On force en cas de suppression ou remplacement.
 - recurse => true : C'est ce qui me permet de dire de copier tout le contenu
 - backup => false : On ne sauvegarde pas les fichiers avant de les remplacer
- Fast reboot command : Je rajoute un fichier exécutable

7.10.3 files

Voici dans files à quoi ressemble mon arborescence dans le dossier files :

```

0. |
1. |-- admin-scripts
2. |   |-- script1.pl
3. |   |-- script2.pl
4. |-- fastreboot

```

Pour information, la commande fastreboot est disponible ici.

7.11 Timezone

Gérer les timezone, c'est plus ou moins facile suivant les OS. Nous allons voir ici comment s'en sortir sur Red Hat. Créons l'arborescence :

```

mkdir

0. mkdir -p /etc/puppet/modules/timezone/{manifests,templates}

```

7.11.1 init.pp

```

/etc/puppet/modules/timezone/manifests/init.pp

```

```

0. /*
1. Timezone Module for Puppet
2. Made by Pierre Mavro
3. */
4. class timezone {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'RedHat' : {
8.       include timezone::redhat
9.     }
10.    #'sunos': { include packages_defaults::solaris }
11.    default : {
12.      notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.    }
14.  }
15. }

```

7.11.2 redhat.pp


Nous utilisons une variable '\$set_timezone' stockée dans les variables globales ou dans la configuration d'un node avec le continent et le pays. Voici un exemple :

```

0. # NTP Timezone. Usage :
1. # Look at /usr/share/zoneinfo/ and add the continent folder followed by the town
2. $set_timezone = 'Europe/Paris'

```

Et le manifest :

 /etc/puppet/modules/timezone/manifests/redhat.pp

```

0. /*
1. Timezone Module for Puppet
2. Made by Pierre Mavro
3. */
4. class timezone::redhat
5. {
6.   # Usage : set a var called set_timezone with required informations. Ex :
7.   # set_timezone = "Europe/Paris"
8.
9.   # Set timezone file
10.  file { ['/etc/sysconfig/clock']:
11.    content => template("timezone/clock.${::operatingsystem}"),
12.    mode => 644,
13.    owner => root,
14.    group => root
15.  }
16.
17.  # Create required Symlink
18.  file { ['/etc/localtime']:
19.    ensure => link,
20.    target => "/usr/share/zoneinfo/${set_timezone}"
21.  }
22. }

```

7.11.3 templates

Et le fichier template nécessaire pour Red Hat :

 /etc/puppet/modules/timezone/templates/clock.RedHat

```

0. ZONE="<<%= set_timezone %>"

```

7.12 NTP

Si vous souhaitez comprendre comment configurer un serveur NTP, je vous invite à lire cette documentation.

Créons l'arborescence :


 mkdir

```

0. mkdir -p /etc/puppet/modules/ntp/{manifests,files}

```

7.12.1 init.pp

 /etc/puppet/modules/ntp/manifests/init.pp

```

0. /*
1. NTP Module for Puppet
2. Made by Pierre Mavro
3. */
4. class ntp {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'RedHat' : {
8.       include ntp::redhat
9.     }
10.    #'sunos': { include packages_defaults::solaris }
11.    default : {

```

```

12.         notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.     }
14. }
15. }

```

7.12.2 redhat.pp

Je souhaite ici que le package soit installé, configuré au boot, qu'il récupère un fichier de configuration classique et qu'il configure la crontab pour que le service ne se lance qu'en dehors des heures de production (pour que les logs de prod aient une cohérence) :

 /etc/puppet/modules/ntp/manifests/redhat.pp

```

0. /*
1. NTP Module for Puppet
2. Made by Pierre Mavro
3. */
4. class ntp::redhat {
5.     # Install NTP service
6.     package {
7.         'ntp' :
8.             ensure => 'installed'
9.     }
10.
11.     # Be sure that service is set to start at boot
12.     service {
13.         'ntpd' :
14.             enable => false
15.     }
16.
17.     # Set configuration file
18.     file {
19.         '/etc/ntp.conf' :
20.             ensure => present,
21.             source => "puppet:///modules/ntp/${::osfamily}.ntp.conf",
22.             mode => 644,
23.             owner => root,
24.             group => root
25.     }
26.
27.     # Enable ntp service during off production hours
28.     cron {
29.         'ntp_start' :
30.             command => '/etc/init.d/ntpd start',
31.             user => root,
32.             minute => 0,
33.             hour => 0
34.     }
35.
36.     # Disable ntp service during on production hours
37.     cron {
38.         'ntp_stop' :
39.             command => '/etc/init.d/ntpd stop',
40.             user => root,
41.             minute => 3,
42.             hour => 0
43.     }
44. }

```

Vous remarquerez l'utilisation des directives 'cron' dans puppet qui permettent le management des lignes de crontab pour un utilisateur donné.

7.12.3 files

Voici le fichier de configuration pour Red Hat :

 /etc/puppet/modules/ntp/files/RedHat.ntp.conf

```

0. # For more information about this file, see the man pages
1. # ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).
2.
3. driftfile /var/lib/ntp/drift
4.
5. # Permit time synchronization with our time source, but do not
6. # permit the source to query or modify the service on this system.
7. restrict default kod nomodify notrap nopeer noquery
8. restrict -6 default kod nomodify notrap nopeer noquery
9.
10. # Permit all access over the loopback interface. This could
11. # be tightened as well, but to do so would effect some of
12. # the administrative functions.
13. restrict 127.0.0.1
14. restrict -6 ::1
15.
16. # Hosts on local network are less restricted.
17. #restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
18.
19. # Use public servers from the pool.ntp.org project.
20. # Please consider joining the pool (http://www.pool.ntp.org/join.html).
21. server 0.rhel.pool.ntp.org
22. server 1.rhel.pool.ntp.org
23. server 2.rhel.pool.ntp.org
24.
25. #broadcast 192.168.1.255 autokey # broadcast server
26. #broadcastclient # broadcast client
27. #broadcast 224.0.1.1 autokey # multicast server
28. #multicastclient 224.0.1.1 # multicast client
29. #manycastserver 239.255.254.254 # manycast server
30. #manycastclient 239.255.254.254 autokey # manycast client
31.
32. # Undisciplined Local Clock. This is a fake driver intended for backup
33. # and when no outside source of synchronized time is available.
34. #server 127.127.1.0 # local clock
35. #fudge 127.127.1.0 stratum 10
36.
37. # Enable public key cryptography.
38. #crypto
39.
40. includefile /etc/ntp/crypto/pw
41.
42. # Key file containing the keys and key identifiers used when operating
43. # with symmetric key cryptography.
44. keys /etc/ntp/keys
45.
46. # Specify the key identifiers which are trusted.
47. #trustedkey 4 8 42
48.

```

```

49. # Specify the key identifier to use with the ntpdc utility.
50. #requestkey 8
51.
52. # Specify the key identifier to use with the ntpq utility.
53. #controlkey 8
54.
55. # Enable writing of statistics records.
56. #statistics clockstats cryptostats loopstats peerstats

```

7.13 MySecureShell

La configuration de MySecureShell n'est pas très complexe, seulement elle diffère généralement d'une machine à une autre, surtout si vous avez un gros parc. Vous souhaitez certainement avoir une gestion identique globale et pouvoir faire du custom sur certains utilisateurs, groupes, virtualhost, etc... C'est pourquoi nous allons gérer ceci de façon la plus simple possible, c'est à dire une configuration globale, puis des includes.

Créons l'arborescence :

```

> mkdir

```

```

0. mkdir -p /etc/puppet/modules/mysecureshell/{manifests,files}

```

7.13.1 init.pp

```

/etc/puppet/modules/mysecureshell/manifests/init.pp

```

```

0. /*
1. MySecureShell Module for Puppet
2. Made by Pierre Mavro
3. */
4. class mysecureshell {
5.     # Check OS and request the appropriate function
6.     case $::operatingsystem {
7.         'RedHat' : {
8.             include mysecureshell::redhat
9.         }
10.         default : {
11.             notice("Module ${module_name} is not supported on ${::operatingsystem}")
12.         }
13.     }
14. }

```

7.13.2 redhat.pp

Ici rien de sorcier, je m'assure que le package est bien installé et la configuration correctement poussée :

```

/etc/puppet/modules/mysecureshell/files/init.pp

```

```

0. /*
1. MySecureShell Module for Puppet
2. Made by Pierre Mavro
3. */
4. class mysecureshell::redhat
5. {
6.     # Install MySecureShell
7.     package { 'mysecureshell':
8.         ensure => 'installed'
9.     }
10.
11.     # MySecureShell default configuration
12.     file {
13.         '/etc/ssh/sftp_config' :
14.             ensure => present,
15.             source => "puppet:///modules/mysecureshell/${::osfamily}.sftp_config",
16.             mode => 644,
17.             owner => root,
18.             group => root
19.         }
20. }

```

7.13.3 files

Voici mon fichier globale, à la fin de celui ci vous noterez qu'il y a un include. Libre à vous de mettre ce que vous voulez dedans, d'en avoir plusieurs et d'envoyer le bon suivant des critères :

```

/etc/puppet/modules/mysecureshell/files/RedHat.sftp_config

```

```

0. # HEADER: This file is managed by puppet.
1. # HEADER: While it can still be managed manually, it is definitely not recommended.
2.
3. #Default rules for everybody
4. <default>
5.     GlobalDownload      0      #total speed download for all clients
6.     GlobalUpload        0      #total speed download for all clients (0 for unlimited)
7.     Download            0      #limit speed download for each connection
8.     Upload              0      #unlimit speed upload for each connection
9.     StayAtHome          true   #limit client to his home
10.    VirtualChroot        true   #fake a chroot to the home account
11.    LimitConnection      100    #max connection for the server sftp
12.    LimitConnectionByUser 2      #max connection for the account
13.    LimitConnectionByIP  2      #max connection by ip for the account
14.    Home                 /home/$USER #override home of the user but if you want you can use
15.    IdleTimeOut          5m     #(in second) deconnect client is idle too long time
16.    ResolveIP            false  #resolve ip to dns

```

```

17. IgnoreHidden true #treat all hidden files as if they don't exist
18. # DirFakeUser true #Hide real file/directory owner (just change displayed permissions)
19. # DirFakeGroup true #Hide real file/directory group (just change displayed permissions)
20. # DirFakeMode 0400 #Hide real file/directory rights (just change displayed permissions)
21. # HideFiles "(lost+found|public_html)$" #Hide file/directory which match
22. # HideNoAccess true #Hide file/directory which user has no access
23. # MaxOpenFilesForUser 20 #limit user to open x files on same time
24. # MaxWriteFilesForUser 10 #limit user to x upload on same time
25. # MaxReadFilesForUser 10 #limit user to x download on same time
26. # DefaultRights 0640 0750 #Set default rights for new file and new directory
27. # MinimumRights 0400 0700 #Set minimum rights for files and dirs
28. # PathDenyFilter "\." #deny upload of directory/file which match this extended POSIX regex
29. # ShowLinksAsLinks false #show links as their destinations
30. # ConnectionMaxLife 1d #limits connection lifetime to 1 day
31. # Charset "ISO-8859-15" #set charset of computer
32. # GMTime +1 #set GMT Time (change if necessary)
33. </Default>
34.
35. # Include another custom file
36. Include /etc/ssh/deimos_ftp_config #include this valid configuration file

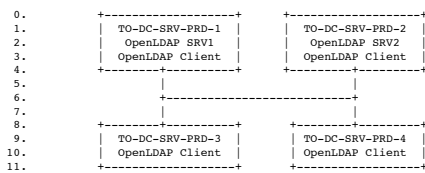
```

7.14 OpenLDAP client & Server

OpenLDAP peut s'avérer vite compliqué et il faut bien comprendre comment il fonctionne avant de le déployer. Je vous invite à regarder les documentations là dessus avant de vous lancer dans ce module.

Ce module est un gros morceau, j'y ai passé pas mal de temps pour qu'il fonctionne de façon complètement automatisée. Je vais être le plus explicite possible pour que tout soit bien clair. Je l'ai adapté pour une nomenclature des noms qui permet de genre de chose dans un parc informatique, mais nous pouvons nous référer à à peut prêt n'importe quoi.

Dans le cas de ce module, pour bien le comprendre, il faut que je vous explique comment est constitué l'infrastructure. Nous pouvons partir du principe que nous avons un lot ou un cluster à 4 noeuds. Sur ces 4 noeuds, seuls les machines numérotées 1 et 2 sont serveurs OpenLDAP. Tous les noeuds sont clients des serveurs :



Créons l'arborescence :



mkdir

```
0. mkdir -p /etc/puppet/modules/openldap/{manifests,files,lib/facter,puppet/parser/functions}
```

7.14.1 init.pp

Je viens charger ici tous mes manifests pour pouvoir en appeler leurs classes par la suite :



/etc/puppet/modules/openldap/manifests/init.pp

```

0. /*
1. OpenLDAP Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openldap inherits openssh {
5.   import "*"
6.
7.   # Check OS and request the appropriate function
8.   case $::operatingsystem {
9.     "RedHat" {
10.      include openldap::redhat
11.     }
12.     # 'sunos': { include packages_defaults::solaris }
13.     default {
14.       notice("Module ${module_name} is not supported on ${::operatingsystem}")
15.     }
16.   }
17. }

```

7.14.2 redhat.pp

Je vais m'occuper ici de toute la gestion OpenLDAP :

- Ligne 6 à 17 : Préparation
- Ligne 20 : Configuration du client
- Ligne 22 à 27 : Configuration du serveur



/etc/puppet/modules/openldap/manifests/redhat.pp

```

0. /*
1. OpenLDAP Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openldap::redhat inherits openldap {
5.   # Select generated ldap servers if no one is specified or join specified ones
6.   if empty("${ldap_servers}") == true
7.   {
8.     $comma_ldap_servers = "${::generated_ldap_servers}"
9.   }

```

```

10. else
11. {
12.   $comma_ldap_servers = inline_template("<$= (ldap_servers).join(',') >")
13. }
14.
15. # DNS lookup them to use IP address instead
16. $ip_ldap_servers = dns2ip("${comma_ldap_servers}")
17.
18. # OpenLDAP Client
19. include openldap::redhat::ldapclient
20.
21. # Check if the current host is a server or not
22. $ip_hostname = dns2ip($:hostname)
23. $array_ldap_servers = split($ip_ldap_servers, ',')
24. if $ip_hostname in $array_ldap_servers {
25.   # OpenLDAP Server
26.   include openldap::redhat::ldapservers
27. }
28. }

```

Nous utilisons une variable '\$ldap_servers' stockée dans les variables globales ou dans la configuration d'un node avec la liste des serveurs LDAP par défaut souhaités :

```

0. # Default LDAP servers
1. $ldap_servers = [ '192.168.0.1', '192.168.0.2' ]
2. $ldap_servers = [ ]

```

Si nous utilisons comme ici, un tableau vide, le parser empty (ligne 7) le détectera et nous utiliserons alors la génération automatique de la liste des serveurs LDAP à utiliser. Cette méthode utilise un facteur pour détecter en fonction du nom de la machine son équivalent 1 et 2.

Ensuite, il fera un reverse lookup pour convertir en IP, les éventuels noms DNS qui figureraient dans la liste des serveurs LDAP. Pour cela, nous utilisons un parser nommé dns2ip. Puis il y aura l'appel du manifest OpenLDAP client.

Et pour finir, si le serveur actuelle figure dans la liste des serveurs OpenLDAP, alors on lui applique sa configuration.

7.14.3 factor

7.14.3.1 generated_ldap_servers.rb

Ce facteur récupère le hostname de la machine en question, si elle correspond à la nomenclature souhaitée, elle envoie dans un tableau les noms auto-générés des serveurs LDAP. Ensuite elle renvoie en résultat le tableau joint par des virgules :

 /etc/puppet/modules/openldap/lib/factor/generated_ldap_servers.rb

```

0. # Generated ldap servers
1. Factor.add("generated_ldap_servers") do
2.   ldap_srv = []
3.
4.   # Get current hostname and add -1 and -2 to define default LDAP servers
5.   hostname = Factor.value('hostname')
6.   if hostname =~ /^(.*)-\d{1,2}$/
7.     ldap_srv.push($1 + '-1', $1 + '-2')
8.   end
9.
10.  setcode do
11.    # Join them with a comma
12.    ldap_srv.join(',')
13.  end
14. end

```

7.14.3.2 current_ldap_servers.rb

Ce facteur lit le fichier de configuration d'OpenLDAP pour voir la configuration qui lui ai appliquée et renvoie le résultat séparé par une virgule :

 /etc/puppet/modules/openldap/lib/factor/current_ldap_servers.rb

```

0. # Current ldap servers
1. Factor.add("current_ldap_servers") do
2.   ldap_srv = []
3.   conf_ldap = []
4.   config_found = 0
5.
6.   # Add in an array all ldap servers lines
7.   f = File.open('/etc/openldap/ldap.conf', 'r')
8.   f.each_line do |line|
9.     if line =~ /^URI (.*)/
10.      config_found = 1
11.      conf_ldap = $1.split(" ")
12.    end
13.  end
14.  f.close
15.
16.  # Get all LDAP servers names/IP
17.  if conf_ldap.each do |line|
18.    if line =~ /ldap:\\/\(.*)\\/
19.      ldap_srv.push($1)
20.    end
21.  end
22. end
23.
24. setcode do
25.   if config_found == 1
26.     # Join them with a comma
27.     ldap_srv.join(',')
28.   else
29.     config_found
30.   end
31. end
32. end

```

7.14.4 Parser

7.14.4.1 dns2ip.rb


Ce parser permet de renvoyer une fois traitées, une liste de machines séparées par des virgules. Le traitement est en fait la conversion de nom DNS en IP :

 /etc/puppet/modules/openldap/lib/puppet/parser/functions/dns2ip.rb

```
0. # Dns2IP for Puppet
1. # Made by Pierre Mavro
2. # Does a DNS lookup and returns an array of strings of the results
3. # Usage : need to send one string dns servers separated by comma. The return will be the same
4.
5. require 'resolv'
6.
7. module Puppet::Parser::Functions
8.   newfunction(:dns2ip, :type => :rvalue) do |arguments|
9.     result = []
10.    # Split comma sperated list in array
11.    dns_array = arguments[0].split(',')
12.    # Push each DNS/IP address in result array
13.    dns_array.each do |dns_name|
14.      result.push(Resolv.new.getaddresses(dns_name))
15.    end
16.    # Join array with comma
17.    dns_list = result.join(',')
18.    # Delete last comma if exist
19.    good_dns_list = dns_list.gsub(/,$/, '')
20.    return good_dns_list
21.  end
22. end
```

7.14.5 redhat_ldapclient.pp

Ce manifest va installer tous les packages nécessaire pour que le serveur devienne client OpenLDAP, puis va s'assurer que 2 services (vous noterez l'utilisation de tableau) tournent et qu'ils sont démarrés au boot. Ensuite on va utiliser les facters précédemment cités, ainsi que d'autres variables custom pour valider que la configuration présente sur le serveur est existante ou non. Dans le cas négatif, nous exécutons une commande qui nous fera la configuration du client OpenLDAP :

 /etc/puppet/modules/openldap/manifests/redhat_ldapclient.pp

```
0. /*
1. OpenLDAP Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openldap::redhat::ldapclient inherits openldap::redhat {
5.   # Install required clients pacakges
6.   packages_install
7.   {
8.     ['nss-pam-ldapd',
9.      'openldap',
10.      'openldap-clients',
11.      'openssh-ldap'
12.     ]
13.   }
14.   # Be sure that service is set to start at boot
15.   service {
16.     ['nscd', 'nslcd'] :
17.       enable => true,
18.       ensure => running,
19.       require => Package['nss-pam-ldapd', 'openldap', 'openldap-clients', 'openssh-ldap']
20.   }
21.   # Configure LDAP client if current configuration doesn't match
22.   if ($::current_ldap_servers != $ip_ldap_servers or $::current_ldap_servers == 0)
23.   {
24.     exec {
25.       "authconfig --enableldap --enableldapauth --disablenis --enablecache --passalgo=sha512 --disableldaptls --disableldapstarttls --disablelsssdauth --enablemkhor
26.       logoutput => true,
27.       require => Service['nslcd']
28.     }
29.   }
30. }
31. }
```

7.14.6 redhat_ldapserver.pp

Pour la partie serveur, nous allons vérifier l'existence de tous les dossiers, vérifier que les packages et services sont correctement configurés, envoyer les schémas customs et la configuration OpenLDAP :

 /etc/puppet/modules/openldap/manifests/redhat_ldapserver.pp

```
0. /*
1. OpenLDAP Module for Puppet
2. Made by Pierre Mavro
3. */
4. class openldap::redhat::ldapserver inherits openldap::redhat {
5.   # Install required clients pacakges
6.   packages_install {
7.     ['openldap-servers'] :
8.   }
9.
10.  # Copy schemas
11.  file {
12.    "/etc/openldap/schema" :
13.      ensure => directory,
14.      mode => 755,
15.      owner => root,
16.      group => root,
17.      source => "puppet:///openldap/schema/",
18.      recurse => true
19.  }
20.
21.  # Copy configuration folder (new format)
22.  file {
23.    "/etc/openldap/slapd.d" :
24.      ensure => directory,
25.      mode => 700,
26.      owner => ldap,
27.      group => ldap,
28.      source => "puppet:///openldap/$::operatingsystem/slapd.d/",
29.      recurse => true
30.  }
31. }
```




```

31.
32. # Copy configuration file (old format)
33. file {
34.     "/etc/openldap/slapd.conf" :
35.         ensure => present,
36.         mode => 700,
37.         owner => ldap,
38.         group => ldap,
39.         source => "puppet:///openldap/$::operatingsystem/slapd.conf"
40.     }
41.
42. # Ensure rights are ok for folder LDAP database
43. file {
44.     "/var/lib/ldap" :
45.         ensure => directory,
46.         mode => 700,
47.         owner => ldap,
48.         group => ldap
49.     }
50.
51. # Ensure rights are ok for folder pid and args
52. file {
53.     "/var/run/openldap" :
54.         ensure => directory,
55.         mode => 755,
56.         owner => ldap,
57.         group => ldap
58.     }
59.
60. # Be sure that service is set to start at boot
61. service {
62.     'slapd' :
63.         enable => true,
64.         ensure => running,
65.         require => File['/etc/openldap/slapd.conf']
66.     }
67. }

```

7.14.7 files

Pour vous donner une idée du contenu de files :

 /etc/puppet/modules/openldap/files/*

```

0. .
1. -- RedHat
2. |-- slapd.conf
3. |-- slapd.d
4. |   |-- cn=config
5. |   |   |-- cn=module(0).ldif
6. |   |   |-- cn=schema
7. |   |   |   |-- cn=(0)core.ldif
8. |   |   |   |-- cn=(1)cosine.ldif
9. |   |   |   |-- cn=(2)nis.ldif
10. |   |   |   |-- cn=(3)inetorgperson.ldif
11. |   |   |   |-- cn=(4)microsoft.ldif
12. |   |   |   |-- cn=(5)microsoft.ldif
13. |   |   |   |-- cn=(6)microsoft.ldif
14. |   |   |   |-- cn=(7)samba.ldif
15. |   |   |   |-- cn=(8)deimos.ldif
16. |   |   |-- cn=schema.ldif
17. |   |   |-- olcDatabase={-1}frontend.ldif
18. |   |   |-- olcDatabase={0}config.ldif
19. |   |   |-- olcDatabase={1}bdb.ldif
20. |   |-- cn=config.ldif

```

Et pour la configuration du service LDAP :

 /etc/puppet/modules/openldap/files/slapd.conf

```

0. # Schema and objectClass definitions
1. include /etc/openldap/schema/core.schema
2. include /etc/openldap/schema/cosine.schema
3. include /etc/openldap/schema/nis.schema
4. include /etc/openldap/schema/inetorgperson.schema
5. include /etc/openldap/schema/microsoft.schema
6. include /etc/openldap/schema/microsoft.sfu.schema
7. include /etc/openldap/schema/microsoft.exchange.schema
8. include /etc/openldap/schema/samba.schema
9. include /etc/openldap/schema/deimos.schema
10.
11.
12. # Where the pid file is put. The init.d script
13. # will not stop the server if you change this.
14. pidfile /var/run/openldap/slapd.pid
15.
16. # List of arguments that were passed to the server
17. argsfile /var/run/openldap/slapd.args
18.
19. # Read slapd.conf(5) for possible values
20. loglevel 0
21.
22. # Where the dynamically loaded modules are stored
23. modulepath /usr/libexec/openldap
24. moduleload back_bdb
25. #moduleload back_meta
26.
27. # The maximum number of entries that is returned for a search operation
28. sizelimit 500
29.
30. # The tool-threads parameter sets the actual amount of cpu's that is used
31. # for indexing.
32. #tool-threads 1
33.
34. #####
35. # Specific Backend Directives for bdb:
36. # Backend specific directives apply to this backend until another
37. # 'backend' directive occurs
38. backend bdb
39.
40. #####
41. # Specific Directives for database #1, of type bdb:
42. # Database specific directives apply to this database until another
43. # 'database' directive occurs
44. database bdb
45.
46. # The base of your directory in database #1

```

```

47. suffix          "dc=openldap,dc=deimos,dc=fr"
48.
49. # rootdn directive for specifying a superuser on the database. This is needed
50. # for syncrepl.
51. rootdn          "cn=admin,dc=openldap,dc=deimos,dc=fr"
52. rootpw         {SSHA}Sh+Yd...
53.
54. # Where the database file are physically stored for database #1
55. directory       "/var/lib/ldap"
56.
57. # For the Debian package we use 2MB as default but be sure to update this
58. # value if you have plenty of RAM
59. dbconfig set_cachesize 0 2097152 0
60.
61. # Number of objects that can be locked at the same time.
62. dbconfig set_lk_max_objects 1500
63. # Number of locks (both requested and granted)
64. dbconfig set_lk_max_locks 1500
65. # Number of lockers
66. dbconfig set_lk_max_lockers 1500
67.
68. # Indexing options for database #1
69. index           objectClass eq
70. index           cn,sn,uid,mail pres,eq,sub
71. index           mailnickname,userprincipalname,proxyaddresses pres,eq,sub
72. index           entryUUID,entryCSN eq
73.
74.
75. # Save the time that the entry gets modified, for database #1
76. lastmod         on
77.
78. access to attrs=userPassword,shadowLastChange
79.   by dn="cn=replica,ou=Gestion Admin,ou=Utilisateurs,dc=openldap,dc=deimos,dc=fr" write
80.   by anonymous auth
81.   by self write
82.   by * none
83.
84. access to *
85.   by dn="cn=replica,ou=Gestion Admin,ou=Utilisateurs,dc=openldap,dc=deimos,dc=fr" write
86.   by * read
87.
88. access to dn.base="" by * read

```

7.15 sudo

Sudo devient vite indispensable lorsque l'on souhaite donner des droits privilégiés à des groupes ou des utilisateurs. Regardez cette documentation si vous avez besoin de plus d'informations.

7.15.1 init.pp

Le fichier init.pp est le cœur de notre module, renseignez le comme ceci pour le moment :

```

/etc/puppet/modules/sudo/manifests/init.pp

0. # sudo.pp
1. class sudo {
2.   # OS detection
3.   $sudoers_file = $operatingsystem ? {
4.     Solaris => "/opt/csw/etc/sudoers",
5.     default => "/etc/sudoers"
6.   }
7.
8.   # Sudoers file declaration
9.   file { ["$sudoers_file"]:
10.     owner => root,
11.     group => root,
12.     mode => 640,
13.     source => $operatingsystem ? {
14.       Solaris => "puppet:///modules/sudo/sudoers.solaris",
15.       default => "puppet:///modules/sudo/sudoers"
16.     }
17.   }
18.
19.   # Symlink for solaris
20.   case $operatingsystem {
21.     Solaris: {
22.       file { ["/opt/csw/bin/sudo"]:
23.         ensure => ["/opt/csw/bin/sudo"]
24.       }
25.     }
26.   }
27. }

```

Je vais essayer d'être clair dans les explications :

- Nous déclarons une classe sudo
- On fait une détection d'OS. Cette déclaration est hérité d'une variable (\$sudoers_file) qui nous permet de déclarer différents path du fichier sudoers.
- Qui comprends un fichier de configuration se trouvant dans (name) /opt/csw/etc/sudoers sur le système cible (pour cette conf, c'est sur Solaris, a vous d'adapter)
- Le fichier doit appartenir au user et groupe root avec les droits 440.
- La source de ce fichier (que nous n'avons pas encore déposé) est disponible (source) à l'adresse puppet:///modules/sudo/sudoers (soit /etc/puppet/modules/sudo/files/sudoers). Il indique tout simplement l'emplacement des fichiers dans votre arborescence puppet qui utilise un mécanisme de système de fichier interne à Puppet.

7.15.1.1 Amélioration d'un module

Éditons le fichier pour lui rajouter quelques demandes

```

/etc/puppet/modules/sudo/manifests/init.pp

0. # sudo.pp
1. class sudo {
2.
3.   # Check if sudo is the latest version
4.   package { sudo: ensure => latest }
5.
6.   # OS detection
7.   $sudoers_file = $operatingsystem ? {
8.     Solaris => "/opt/csw/etc/sudoers",
9.     default => "/etc/sudoers"
10.  }

```

```

11.
12. # Sudoers file declaration
13. file { "$sudoers_file":
14.     owner => root,
15.     group => root,
16.     mode  => 640,
17.     source => $operatingsystem ? {
18.         Solaris => "puppet:///modules/sudo/sudoers.solaris",
19.         default => "puppet:///modules/sudo/sudoers"
20.     },
21.     require => Package["sudo"]
22. }
23. }

```

- `require => Package["sudo"]` : on lui demande d'installer le package sudo s'il n'est pas installé
- `package { sudo: ensure => latest }` : on lui demande de vérifier que c'est bien la dernière version

7.15.2 files

Maintenant, il faut ajouter les fichiers que nous souhaitons publier dans le dossier files (ici seulement sudoers) :



cp

```

cp /etc/sudoers /etc/puppet/modules/sudo/files/sudoers
cp /etc/sudoers /etc/puppet/modules/sudo/files/sudoers.solaris

```

Pour la faire simple, j'ai simplement copier le sudoers du serveur vers la destination qui correspond à ma conf décrite plus haut.

Si vous n'avez pas installé sudo sur le serveur, mettez un fichier sudoers qui vous convient dans `/etc/puppet/modules/sudo/files/sudoers`.

Dans les prochaines version, puppet supportera le protocole http et ftp pour aller chercher ces fichiers.

7.16 snmpd

Le service snmpd est assez simple d'utilisation. C'est pourquoi ce module aussi :-). Créons l'arborescence :



mkdir

```

0. mkdir -p /etc/puppet/modules/snmpd/{manifests,files}

```

7.16.1 init.pp



/etc/puppet/modules/snmpd/manifests/init.pp

```

0. /*
1. Snmpd Module for Puppet
2. Made by Pierre Mavro
3. */
4. class snmpd {
5.     # Check OS and request the appropriate function
6.     case $::operatingsystem {
7.         'RedHat' : {
8.             include snmpd::redhat
9.         }
10.        #'sunos' : { include packages_defaults::solaris }
11.        default : {
12.            notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.        }
14.    }
15. }

```

7.16.2 redhat.pp



/etc/puppet/modules/snmpd/manifests/redhat.pp


```

0. /*
1. Snmpd Module for Puppet
2. Made by Pierre Mavro
3. */
4. class snmpd::redhat {
5.     # Install snmp packages
6.     package {
7.         'net-snmp':
8.             ensure => present
9.     }
10.
11.     # Snmpd config file
12.     file {
13.         '/etc/snmp/snmpd.conf':
14.             source => "puppet:///modules/snmpd/snmpd.conf.${::operatingsystem}",
15.             mode => 644,
16.             owner => root,
17.             group => root,
18.             notify => Service["snmpd"]
19.     }
20.
21.     # Service should start on boot and be running
22.     service {
23.         'snmpd':
24.             enable => true,
25.             ensure => running,
26.             require => File['/etc/snmp/snmpd.conf']
27.     }
28. }

```

7.16.3 files

Et le fichier de configuration :

 /etc/puppet/modules/snmpd/files/snmpd.conf.RedHat

```

0. # Generated by Puppet
1. #####
2. #
3. # snmpd.conf
4. #
5. # - created by the snmpconf configuration program
6. #
7. #
8. #####
9. # SECTION: System Information Setup
10. #
11. # This section defines some of the information reported in
12. # the "system" mib group in the mibII tree.
13. #
14. # syslocation: The [typically physical] location of the system.
15. # Note that setting this value here means that when trying to
16. # perform an snmp SET operation to the sysLocation.0 variable will make
17. # the agent return the "notWritable" error code. IE, including
18. # this token in the snmpd.conf file will disable write access to
19. # the variable.
20. # arguments: location_string
21. #
22. syslocation Unknown (edit /etc/snmp/snmpd.conf)
23. #
24. # syscontact: The contact information for the administrator
25. # Note that setting this value here means that when trying to
26. # perform an snmp SET operation to the sysContact.0 variable will make
27. # the agent return the "notWritable" error code. IE, including
28. # this token in the snmpd.conf file will disable write access to
29. # the variable.
30. # arguments: contact_string
31. #
32. syscontact Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
33. #
34. #####
35. # SECTION: Access Control Setup
36. #
37. # This section defines who is allowed to talk to your running
38. # snmp agent.
39. #
40. # rocommunity: a SNMPv1/SNMPv2c read-only access community name
41. # arguments: community [default|hostname|network|bits] [oid]
42. #
43. rocommunity lookdeimos
44. #
45. #
46. # Unknown directives read in from other files by snmpconf
47. #
48. com2sec notConfigUser default public
49. group notConfigGroup v1 notConfigUser
50. group notConfigGroup v2c notConfigUser
51. view systemview included .1.3.6.1.2.1.1
52. view systemview included .1.3.6.1.2.1.25.1.1
53. access notConfigGroup "" any noauth exact systemview none none
54. dontLogTCPWrappersConnects yes

```

7.17 Postfix

Pour postfix, nous allons utiliser des templates pour faciliter certaines configurations. Si vous avez besoin de plus d'informations sur comment configurer Postfix, je vous conseil ces documentations.

7.17.1 init.pp

 /etc/puppet/modules/postfix/manifests/init.pp

```

0. /*
1. Postfix Module for Puppet
2. Made by Pierre Mavro
3. */
4. class postfix {
5.     # Check OS and request the appropriate function
6.     case $::operatingsystem {
7.         'RedHat' : {
8.             include postfix::redhat
9.         }

```

```

10.     #'sunos': { include packages_defaults::solaris }
11.     default : {
12.         notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.     }
14. }
15. }

```

7.17.2 redhat.pp

J'utilise ici des transport map qui nécessite d'être reconstruite si leur configuration change. C'est pourquoi nous utilisons 'Subscribe' dans la directive 'Exec' :

 /etc/puppet/modules/postfix/manifests/redhat.pp

```

0. /*
1. Postfix Module for Puppet
2. Made by Pierre Mavro
3. */
4. class postfix::redhat {
5.     # Install postfix packages
6.     package {
7.         'postfix' :
8.             ensure => present
9.     }
10.
11.     # Postfix main config file
12.     file {
13.         '/etc/postfix/main.cf' :
14.             content => template("postfix/main.cf.${::operatingsystem}"),
15.             mode => 644,
16.             owner => root,
17.             group => root,
18.             notify => Service["postfix"]
19.     }
20.
21.     # Postfix transport map
22.     file {
23.         "/etc/postfix/transport" :
24.             source => "puppet:///modules/postfix/transport",
25.             mode => 644,
26.             owner => root,
27.             group => root,
28.             notify => Service["postfix"]
29.     }
30.
31.     # Rebuild the transport map
32.     exec {
33.         'build_transport_map' :
34.             command => "/usr/sbin/postmap /etc/postfix/transport",
35.             subscribe => File["/etc/postfix/transport"],
36.             refreshonly => true
37.     }
38.
39.     # Service should start on boot and be running
40.     service {
41.         'postfix' :
42.             enable => true,
43.             ensure => running,
44.             require => File['/etc/postfix/main.cf']
45.     }
46. }

```

7.17.3 templates

Ici j'ai mon fqdn qui est unique en fonction de la machine :

 /etc/puppet/modules/postfix/manifests/templates/main.cf.RedHat

```

0. # Generated by Puppet
1.
2. # Postfix directories
3. queue_directory = /var/spool/postfix
4. command_directory = /usr/sbin
5. daemon_directory = /usr/libexec/postfix
6. data_directory = /var/lib/postfix
7. sendmail_path = /usr/sbin/sendmail.postfix
8. newaliases_path = /usr/bin/newaliases.postfix
9. mailq_path = /usr/bin/mailq.postfix
10.
11. # Inet configuration
12. inet_interfaces = all
13. inet_protocols = all
14.
15. # Reject unknown recipients
16. unknown_local_recipient_reject_code = 550
17.
18. # Do not set relayhost. Postfix must use transport_maps
19. relayhost =
20.
21. # Destinations
22. mydomain = deimos.fr
23. myorigin = <%= fqdn %>
24. mydestination = $myorigin, localhost.$mydomain, localhost
25.
26. # Transport_maps permits to route email using Google exchangers
27. transport_maps = dnm:/etc/opt/csw/postfix/transport
28. # Add TLS to emails if possible
29. smtp_tls_security_level = may
30.
31. # Masquerade_domains hides hostnames from addresses
32. masquerade_domains = $mydomain
33.
34. # Aliases
35. alias_maps = hash:/etc/aliases
36. alias_database = hash:/etc/aliases
37.
38. # SMTP banner
39. smtpd_banner = $myhostname ESMTPEX $mail_name
40.
41. # Debug
42. debug_peer_level = 2
43. debugger_command =
44.     PATH=/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin
45.     ddd $daemon_directory/$process_name $process_id & sleep 5
46.
47. # Postfix rights
48. mail_owner = postfix

```

```

49. setgid_group = postdrop
50.
51. # Helps
52. html_directory = no
53. manpage_directory = /usr/share/man
54. sample_directory = /usr/share/doc/postfix-2.6.6/samples
55. readme_directory = /usr/share/doc/postfix-2.6.6/README_FILES

```

7.17.4 files

Et dans mes files, j'ai mon fichier de transport_map :

 /etc/puppet/modules/postfix/manifests/files/transport

```

0. deimos.fr      :google.com
1. .deimos.fr    :google.com

```

7.18 Nsswitch

Pour nsswitch, on va utiliser une technique avancée qui consiste à utiliser Factor (un built in qui permet de gagner beaucoup de temps). Factor propose d'avoir en gros des variables d'environnements spécifiques à Puppet qui permettent de faire des conditions suite à cela. Par exemple, je souhaite vérifier la présence d'un fichier qui va m'indiquer si mon serveur est en mode cluster (pour Sun Cluster) ou non et modifier le fichier nsswitch en fonction. Pour cela je vais donc utiliser factor.


Créons l'arborescence :

 mkdir

```
mkdir -p /etc/puppet/modules/nsswitch/{manifests,lib/factor}
```

7.18.1 factor

Créons notre fichier de fact :

 /etc/puppet/modules/nsswitch/lib/factor/is_cluster.rb


```

0. # is_cluster.rb
1.
2. Factor.add("is_cluster") do
3.   setcode do
4.     #!x{/bin/uname -i}.chomp
5.     FileTest.exists?("/etc/cluster/nodeid")
6.   end
7. end

```

7.18.2 init.pp

Ensuite nous allons spécifier l'utilisation d'un template :

 /etc/puppet/modules/nsswitch/manifests/init.pp

```

0. class nsswitch {
1.   case $operatingsystem {
2.     Solaris: { include nsswitch::solaris }
3.     default: { }
4.   }
5. }
6.
7. class nsswitch::solaris {
8.   $nsfilename = $operatingsystem ? {
9.     Solaris => "/etc/nsswitch.conf",
10.    default => "/etc/nsswitch.conf"
11.   }
12.
13.   config_file { "$nsfilename":
14.     content => template("nsswitch/nsswitch.conf"),
15.   }
16. }

```

7.18.3 templates

Et maintenant le fichier de conf qui va faire appel au factor :

 /etc/puppet/modules/nsswitch/templates/nsswitch.conf

```

#
# Copyright 2006 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
#
# /etc/nsswitch.dns:
#
# An example file that could be copied over to /etc/nsswitch.conf; it uses
# DNS for hosts lookups, otherwise it does not use any other naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.

```

```

# DNS service expects that an instance of svc:/network/dns/client be
# enabled and online.
#
passwd: files ldap
group: files ldap
# You must also set up the /etc/resolv.conf file for DNS name
# server lookup. See resolv.conf(4).
#hosts: files dns
hosts: <| if is_cluster -|>cluster<| end -|> files dns
# Note that IPv4 addresses are searched for in all of the ipnodes databases
# before searching the hosts databases.
#ipnodes: files dns
#ipnodes: files dns [TRYAGAIN=0]
ipnodes: files dns [TRYAGAIN=0 ]
#
networks: files
protocols: files
rpe: files
rpe: files
rpe: files
#netmasks: files
netmasks: <| if is_cluster -|>cluster<| end -|> files
bootparams: files
publickey: files
# At present there isn't a 'files' backend for netgroup: the system will
# figure it out pretty quickly, and won't use netgroups at all.
netgroup: files
autocount: files
aliases: files
services: files
printers: user files
#
auth_attr: files
prof_attr: files
project: files
#
tntarhttp: files
tntarhdb: files
#

```

7.19 Nagios NRPE + plugins

Vous devez initialiser le module avant de continuer.

7.19.1 init.pp

Imaginons que j'ai un dossier de plugins nagios que je souhaite déployer partout. Des fois j'en ajoute, j'en enlève, bref, je fais ma vie avec et je veux que ce soit pleinement synchronisé. Voici la marche à suivre :

```

/etc/puppet/modules/nrpe/manifests/init.pp

0. #nagios_plugins.pp
1. class nrpe {
2.     # Copy conf and send checks
3.     include nrpe::common
4.
5.     # Check operating system
6.     case $operatingsystem {
7.         Solaris: {
8.             include nrpe::service::solaris
9.         }
10.        default: { }
11.    }
12. }
13.
14. class nrpe::common {
15.     class nrpe::configs {
16.         # Used for nrpe-deimos.cfg templates
17.         $nrpe_distrib = $operatingsystem ? {
18.             'Solaris' => "/opt/csw/libexec/nagios-plugins",
19.             'redhat' => "/home/deimos/checks/nrpe_scripts",
20.             'debian' => "/etc/nrpe",
21.         }
22.
23.         # Used for nrpe-deimos.cfg templates
24.         $deimos_script = $operatingsystem ? {
25.             'Solaris' => "/opt/deimos/libexec",
26.             'redhat' => "/etc/nrpe.d",
27.             'debian' => "/etc/nrpe",
28.         }
29.
30.         # Copy NRPE config file
31.         file { [ '/opt/csw/etc/nrpe.cfg':
32.             mode => 644, owner => root, group => root,
33.             content => template('nrpe/nrpe.cfg'),
34.         ]
35.
36.         ## Copy and adapt NRPE deimos Config file ##
37.         file { [ '/opt/csw/etc/nrpe-deimos.cfg':
38.             mode => 644, owner => root, group => root,
39.             content => template('nrpe/nrpe-deimos.cfg'),
40.         ]
41.     }
42.
43.     class nrpe::copy_checks {
44.         ## Copy deimos Production NRPE Checks ##
45.
46.         file { [ "nrpe_prod_checks":
47.             name => $operatingsystem ? {
48.                 'Solaris' => "/opt/deimos/libexec",
49.                 'redhat' => "/home/deimos/checks/nrpe_scripts",
50.                 'debian' => "/etc/nrpe",
51.             },
52.             ensure => directory,
53.             mode => 755, owner => root, group => root,
54.             source => $operatingsystem ? {
55.                 'Solaris' => [ "puppet:///modules/nrpe/Nagios/nrpechecks/deimos", "puppet:///modules/nrpe/Nagios/nrpechecks/system" ],
56.                 'redhat' => [ "puppet:///modules/nrpe/Nagios/nrpechecks/deimos", "puppet:///modules/nrpe/Nagios/nrpechecks/system" ],
57.                 'debian' => [ "puppet:///modules/nrpe/Nagios/nrpechecks/deimos", "puppet:///modules/nrpe/Nagios/nrpechecks/system" ],
58.             },
59.             force => true,
60.             ignore => '.svn'
61.         ]
62.
63.         include nrpe::configs
64.         include nrpe::copy_checks
65.     }
66.
67.     class nrpe::service::solaris {
68.         ## Check service and restart if needed
69.         file { [ '/var/svc/manifest/network/nagios-nrpe.xml':
70.             source => "puppet:///modules/nrpe/nagios-nrpe.xml",
71.             mode => 644, owner => root, group => sys,
72.             before => Service["svc:/network/cswnrpe:default"]
73.         ]
74.
75.         # Restart service if smf xml has changed
76.         exec { [ "svccfg import /var/svc/manifest/network/nagios-nrpe.xml" :
77.             subscribe => File[ '/var/svc/manifest/network/nagios-nrpe.xml' ],

```

```

78.     refreshonly => true
79.   }
80.
81.   # Restart if one of those files have changed
82.   service { "svc:/network/cswnrpe:default":
83.     ensure => running,
84.     manifest => "/var/svc/manifest/network/nagios-nrpe.xml",
85.     subscribe => [ File["/opt/csw/etc/nrpe.cfg"], File["/opt/csw/etc/nrpe-deimos.cfg"] ]
86.   }
87. }

```

Ici :

- purge permet d'effacer les fichiers qui n'existent plus
- recurse : récursif
- force : permet de forcer
- before : permet d'être exécuter avant autre chose
- subscribe : inscrit une dépendance par rapport à la valeur de celui ci
- refreshonly : rafraichit seulement si il y a des changements

Avec le subscribe, vous pouvez le voir ici, on fait des listes de cette façon : [élément_1, élément_2, élément_3...]. Par contre, petite précision, la changement opère (ici un restart) **seulement si l'un des éléments dans la liste est modifié et non tous.**

Vous pouvez voir également de la ligne 54 à 56, on fait du multi sourcing qui nous permet de spécifier plusieurs sources et en fonction de celles ci d'envoyer à un endroit le contenu de ces divers fichiers.

7.19.2 files

Pour la partie file, j'ai linké avec un svn externe qui permet de m'affranchir de l'intégration des plugins dans la partie puppet (qui entre nous n'a rien à faire ici).

7.19.3 templates

Il suffit de copier le dossier nagios_plugins dans files et de faire les templates ici (je ne ferais que le nrpe.cfg) :

```

/opt/csw/etc/nrpe.cfg
....
command[check_load]=<%= nrpe_distrib %>/check_load -w 15,10,5 -c 30,25,20
command[sunos_check_rss_mem]=<%= deimos_script %>/check_rss_mem.pl -w $ARG1$ -c $ARG2$
....

```

7.20 Munin

Disclaimer : this work is mostly based upon DavidS work, available on his [git repo (<http://git.black.co.at/>)]. In the scope of my work I needed to have munin support for freeBSD & Solaris. I also wrote a class for snmp_plugins & custom plugins. Some things are quite dependant from my infrastructure, like munin.conf generation script but it can easily be adapted to yours, by extracting data from your CMDB.

It requires the munin_interfaces fact published here (and merged into DavidS repo, thanks to him), and [Volcane's extlookup function (<http://nephilim.ml.org/~rip/puppet/extlookup.rb>)] to store some parameters. Enough talking, this is the code :

```

0. # Munin config class
1. # Many parts taken from David Schmitt's http://git.black.co.at/
2. # FreeBSD & Solaris + SNMP & custom plugins support by Nicolas Szalay <nico@ygu.info>
3.
4. class munin::node {
5.   case $operatingsystem {
6.     openbsd: {
7.       debian: { include munin::node::debian }
8.       freebsd: { include munin::node::freebsd }
9.       solaris: { include munin::node::solaris }
10.      default: {}
11.    }
12.  }
13.
14. class munin::node::debian {
15.
16.   package { ["munin-node": ensure => installed ]
17.
18.   file {
19.     "/etc/munin":
20.       ensure => directory,
21.       mode => 0755,
22.       owner => root,
23.       group => root;
24.
25.     "/etc/munin/munin-node.conf":
26.       source => "puppet://$fileserver/files/apps/munin/munin-node-debian.conf",
27.       owner => root,
28.       group => root,
29.       mode => 0644,
30.       before => Package["munin-node"],
31.       notify => Service["munin-node"],
32.     }
33.
34.     service { "munin-node": ensure => running }
35.
36.     include munin::plugins::linux
37.   }
38.
39. class munin::node::freebsd {
40.   package { ["munin-node": ensure => installed, provider => freebsd ]
41.
42.   file { "/usr/local/etc/munin/munin-node.conf":
43.     source => "puppet://$fileserver/files/apps/munin/munin-node-freebsd.conf",
44.     owner => root,
45.     group => wheel,
46.     mode => 0644,
47.     before => Package["munin-node"],
48.     notify => Service["munin-node"],
49.   }
50.
51.   service { "munin-node": ensure => running }
52.
53.   include munin::plugins::freebsd
54. }

```



```

55.
56. class munin::node::solaris {
57.   # "hand made" install, no package.
58.   file { ["/etc/munin/munin-node.conf":
59.     source => "puppet://$fileservers/files/apps/munin/munin-node-solaris.conf",
60.     owner => root,
61.     group => root,
62.     mode => 0644,
63.   ]
64.
65.   include munin::plugins::solaris
66. }
67.
68. class munin::gatherer {
69.   package { [ "munin":
70.     ensure => installed
71.   ]
72.
73.   # custom version of munin-graph : forks & generates many graphs in parallel
74.   file { [ "/usr/share/munin/munin-graph":
75.     owner => root,
76.     group => root,
77.     mode => 0755,
78.     source => "puppet://$fileservers/files/apps/munin/gatherer/munin-graph",
79.     require => Package["munin"]
80.   ]
81.
82.   # custom version of debian cron file. Month & Year cron are generated once daily
83.   file { [ "/etc/cron.d/munin":
84.     owner => root,
85.     group => root,
86.     mode => 0644,
87.     source => "puppet://$fileservers/files/apps/munin/gatherer/munin.cron",
88.     require => Package["munin"]
89.   ]
90.
91.   # Ensure cron is running, to fetch every 5 minutes
92.   service { [ "cron":
93.     ensure => running
94.   ]
95.
96.   # Ruby DBI for mysql
97.   package { [ "libdbd-mysql-ruby":
98.     ensure => installed
99.   ]
100.
101.   # config generator
102.   file { [ "/opt/scripts/muningen.rb":
103.     owner => root,
104.     group => root,
105.     mode => 0755,
106.     source => "puppet://$fileservers/files/apps/munin/gatherer/muningen.rb",
107.     require => Package["munin", "libdbd-mysql-ruby"]
108.   ]
109.
110.   # regenerate munin's gatherer config every hour
111.   cron { [ "munin_config":
112.     command => "/opt/scripts/muningen.rb > /etc/munin/munin.conf",
113.     user => "root",
114.     minute => "0",
115.     require => File["/opt/scripts/muningen.rb"]
116.   ]
117.
118.   include munin::plugins::snmp
119.   include munin::plugins::linux
120.   include munin::plugins::custom::gatherer
121. }
122.
123.
124. # define to create a munin plugin inside the right directory
125. define munin::plugin ($ensure = "present") {
126.
127.   case $operatingsystem {
128.     freebsd: {
129.       $script_path = "/usr/local/share/munin/plugins"
130.       $plugins_dir = "/usr/local/etc/munin/plugins"
131.     }
132.     debian: {
133.       $script_path = "/usr/share/munin/plugins"
134.       $plugins_dir = "/etc/munin/plugins"
135.     }
136.     solaris: {
137.       $script_path = "/usr/local/munin/lib/plugins"
138.       $plugins_dir = "/etc/munin/plugins"
139.     }
140.     default: { }
141.   }
142.
143.   $plugin = "$plugins_dir/$name"
144.
145.   case $ensure {
146.     "absent": {
147.       debug ( "munin_plugin: suppressing $plugin" )
148.       file { $plugin: ensure => absent, }
149.     }
150.
151.     default: {
152.       $plugin_src = $ensure ? { "present" => $name, default => $ensure }
153.
154.       file { $plugin:
155.         ensure => "$script_path/$plugin_src",
156.         require => Package["munin-node"],
157.         notify => Service["munin-node"],
158.       }
159.     }
160.   }
161. }
162.
163. # snmp plugin define, almost same as above
164. define munin::snmp_plugin ($ensure = "present") {
165.   $pluginname = get_plugin_name($name)
166.
167.   case $operatingsystem {
168.     freebsd: {
169.       $script_path = "/usr/local/share/munin/plugins"
170.       $plugins_dir = "/usr/local/etc/munin/plugins"
171.     }
172.     debian: {
173.       $script_path = "/usr/share/munin/plugins"
174.       $plugins_dir = "/etc/munin/plugins"
175.     }
176.     solaris: {
177.       $script_path = "/usr/local/munin/lib/plugins"
178.       $plugins_dir = "/etc/munin/plugins"
179.     }
180.     default: { }
181.   }
182.
183.   $plugin = "$plugins_dir/$name"
184.
185.   case $ensure {
186.     "absent": {
187.       debug ( "munin_plugin: suppressing $plugin" )
188.       file { $plugin: ensure => absent, }
189.     }
190.
191.     "present": {

```

```

192.         file { $plugin:
193.             ensure => "$script_path/${pluginname}",
194.             require => Package["munin-node"],
195.             notify => Service["munin-node"],
196.         }
197.     }
198. }
199. }
200.
201. class munin::plugins::base
202. {
203.     case $operatingsystem {
204.         debian: { $plugins_dir = "/etc/munin/plugins" }
205.         freebsd: { $plugins_dir = "/usr/local/etc/munin/plugins" }
206.         solaris: { $plugins_dir = "/etc/munin/plugins" }
207.         default: {}
208.     }
209.
210.     file { $plugins_dir:
211.         source => "puppet://$fileserver/files/empty",
212.         ensure => directory,
213.         checksum => mtime,
214.         ignore => ".svn*",
215.         mode => 0755,
216.         recurse => true,
217.         purge => true,
218.         force => true,
219.         owner => root
220.     }
221. }
222.
223. class munin::plugins::interfaces
224. {
225.     $ifs = gsub(split($munin_interfaces, " "), "(.+)", "if_\\1")
226.     $if_errs = gsub(split($munin_interfaces, " "), "(.+)", "if_err_\\1")
227.     plugin {
228.         $ifs: ensure => "if_";
229.         $if_errs: ensure => "if_err_";
230.     }
231.
232.     include munin::plugins::base
233. }
234.
235. class munin::plugins::linux
236. {
237.     plugin { [ cpu, load, memory, swap, irq_stats, df, processes, open_files, ntp_offset, vmstat ]:
238.         ensure => "present"
239.     }
240.
241.     include munin::plugins::base
242.     include munin::plugins::interfaces
243. }
244.
245. class munin::plugins::nfsclient
246. {
247.     plugin { "nfs_client":
248.         ensure => present
249.     }
250. }
251.
252. class munin::plugins::snmp
253. {
254.     # initialize plugins
255.     $snmp_plugins=extlookup("munin_snmp_plugins")
256.     snmp_plugin { $snmp_plugins:
257.         ensure => present
258.     }
259.
260.     # SNMP communities used by plugins
261.     file { ["/etc/munin/plugin-conf.d/snmp_communities":
262.         owner => root,
263.         group => root,
264.         mode => 0644,
265.         source => "puppet://$fileserver/files/apps/munin/gatherer/snmp_communities"
266.     ]
267. }
268. }
269.
270. define munin::custom_plugin($ensure = "present", $location = "/etc/munin/plugins") {
271.     $plugin = "$location/$name"
272.
273.     case $ensure {
274.         "absent": {
275.             file { $plugin: ensure => absent, }
276.         }
277.
278.         "present": {
279.             file { $plugin:
280.                 owner => root,
281.                 mode => 0755,
282.                 source => "puppet://$fileserver/files/apps/munin/custom_plugins/$name",
283.                 require => Package["munin-node"],
284.                 notify => Service["munin-node"],
285.             }
286.         }
287.     }
288. }
289.
290. class munin::plugins::custom::gatherer
291. {
292.     $plugins=extlookup("munin_custom_plugins")
293.     custom_plugin { $plugins:
294.         ensure => present
295.     }
296. }
297.
298. class munin::plugins::freebsd
299. {
300.     plugin { [ cpu, load, memory, swap, irq_stats, df, processes, open_files, ntp_offset, vmstat ]:
301.         ensure => "present",
302.     }
303.
304.     include munin::plugins::base
305.     include munin::plugins::interfaces
306. }
307.
308. class munin::plugins::solaris
309. {
310.     # Munin plugins on solaris are quite ... buggy. Will need rewrite / custom plugins.
311.     plugin { [ cpu, load, netstat ]:
312.         ensure => "present",
313.     }
314.
315.     include munin::plugins::base
316.     include munin::plugins::interfaces
317. }

```

7.20.1 Munin Interfaces

Everyone using puppet knows DavidS awesome git repository : git.black.co.at. Unfortunately for me, his puppet infrastructure seems to be almost only linux based. I have different OS in

mine, including FreeBSD & OpenSolaris. Looking at his module-munin I decided to reuse it (and not recreate the wheel) but he used a custom fact that needed some little work. So this is a FreeBSD & (Open)Solaris capable version, to know what network interfaces have link up.

```

0. # return the set of active interfaces as an array
1. # taken from http://git.black.co.at
2. # modified by nico <nico@gcu.info> to add FreeBSD & Solaris support
3.
4. Facter.add("munin_interfaces") do
5.
6.   setcode do
7.     # linux
8.     if Facter.value('kernel') == "Linux" then
9.       `ip -o link show`.split(/\n/).collect do |line|
10.        value = nil
11.        matches = line.match(/^d*: ([^:]*): <(.*,)?UP(,.*?)?>/)
12.        if !matches.nil?
13.          value = matches[1]
14.          value.gsub!(/%.*%, ' ')
15.        end
16.        value
17.      end.compact.sort.join(" ")
18.    #end
19.
20.    # freebsd
21.    elsif Facter.value('kernel') == "FreeBSD" then
22.      Facter.value('interfaces').split(/,/).collect do |interface|
23.        status = `ifconfig #{interface} | grep status`
24.        if status != "" then
25.          status=status.strip!.split(":")[1].strip!
26.          if status == "active" then # I CAN HAZ LINK ?
27.            interface.to_a
28.          end
29.        end
30.      end.compact.sort.join(" ")
31.    #end
32.
33.    # solaris
34.    elsif Facter.value('kernel') == "SunOS" then
35.      Facter.value('interfaces').split(/,/).collect do |interface|
36.        if interface != "lo0" then # /dev/lo0 does not exists
37.          status = `ndd -get /dev/#{interface} link_status`.strip!
38.          if status == "1" # ndd returns 1 for link up, 0 for down
39.            interface.to_a
40.          end
41.        end
42.      end.compact.sort.join(" ")
43.    end
44.  end
45. end

```

7.21 Mcollective

Mcollective est un outil je l'on couple généralement à Puppet pour améliorer notre quotidien. Si vous ne connaissez pas ou voulez en apprendre plus, suivez ce lien.

Ce module Mcollective pour Puppet permet d'installer mcollective, ainsi que des modules côté client (serveurs Mcollective). Avec les modules, voici à quoi ressemble mon arborescence :

```

-- files
|-- agent
|   |-- filemgr.rb
|   |-- nrpe.rb
|   |-- package.rb
|   |-- process.rb
|   |-- puppetd.rb
|   |-- service.rb
|   |-- shell.rb
-- facts
   |-- factor facts.rb
   |-- Redhat.server.cfg
-- manifests
   |-- common.pp
   |-- init.pp
   |-- redhat.pp

```

Pour ce qui est des modules (agents,facts), je vous invite à regarder ma doc sur Mcollective pour savoir où les récupérer. Créons l'arborescence :

```

0. mkdir -p /etc/puppet/modules/mcollective/(manifests,files)

```

7.21.1 init.pp

Le fichier init.pp est le cœur de notre module, renseignez le comme ceci :


```

/etc/puppet/modules/mcollective/manifests/init.pp

0. /*
1. Mcollective Module for Puppet
2. Made by Pierre Mavro
3. */
4. class mcollective {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'Redhat' : {
8.       include mcollective::redhat
9.     }
10.    default : {
11.      notice("Module ${module_name} is not supported on ${::operatingsystem}")
12.    }
13.  }
14. }

```

7.21.2 common.pp

 /etc/puppet/modules/mcollective/manifests/common.pp

```
0. /*
1. Mcollective Module for Puppet
2. Made by Pierre Mavro
3. */
4. class mcollective::common {
5.     # Used for nrpe.cfg file
6.     $mco_agent_dir = $operatingsystem ? {
7.         'Redhat' => "/usr/libexec/mcollective/mcollective",
8.         'Solaris' => "/usr/libexec/mcollective/mcollective"
9.     }
10.
11.     # Transfert agents
12.     file {
13.         'mco_agent_folder' :
14.             name => "$mco_agent_dir/agent",
15.             ensure => directory,
16.             mode => 644,
17.             owner => root,
18.             group => root,
19.             source => ["puppet:///modules/mcollective/agent"],
20.             recurse => true,
21.             ignore => '.svn',
22.             backup => false,
23.             notify => Service['mcollective'],
24.             require => Package['mcollective']
25.     }
26.
27.     # Transfert facts
28.     file {
29.         'mco_facts_folder' :
30.             name => "$mco_agent_dir/facts",
31.             ensure => directory,
32.             mode => 644,
33.             owner => root,
34.             group => root,
35.             source => ["puppet:///modules/mcollective/facts"],
36.             recurse => true,
37.             ignore => '.svn',
38.             backup => false,
39.             notify => Service['mcollective'],
40.             require => Package['mcollective']
41.     }
42. }
```

7.21.3 redhat.pp

 /etc/puppet/modules/mcollective/manifests/redhat.pp

```
0. /*
1. Mcollective Module for Puppet
2. Made by Pierre Mavro
3. */
4. class mcollective::redhat {
5.     # Install Mcollective client
6.     package { [
7.         'mcollective',
8.         'rubygem-stomp',
9.         'rubygem-sysproctable'
10.     ]:
11.         ensure => 'installed'
12.     }
13.
14.     # Be sure that service is set to start at boot and running
15.     service {
16.         'mcollective' :
17.             enable => true,
18.             ensure => running
19.     }
20.
21.     # Set configuration file
22.     file {
23.         '/etc/mcollective/server.cfg' :
24.             ensure => present,
25.             source => "puppet:///modules/mcollective/${::osfamily}.server.cfg",
26.             mode => 640,
27.             owner => root,
28.             group => root,
29.             notify => Service['mcollective']
30.     }
31.
32.     # Include common
33.     include mcollective::common
34. }
```

7.21.4 files

7.21.4.1 RedHat.server.cfg

Voici la configuration pour Mcollective :

 /etc/mcollective/server.cfg

```
0. #####
1. # GLOBAL CONFIGURATION #
2. #####
3.
4. topicprefix = /topic/
5. main_collective = mcollective
6. collectives = mcollective
7. libdir = /usr/libexec/mcollective
8. logfile = /var/log/mcollective.log
9. loglevel = info
10. daemionise = 1
11. classesfile = /var/lib/puppet/classes.txt
```

```

12.
13. #####
14. # MODULES #
15. #####
16.
17. # Security
18. securityprovider = psk
19. plugin.psk = unset
20.
21. # Stomp
22. connector = stomp
23. plugin.stomp.host = mcollective.deimos.fr
24. plugin.stomp.port = 61613
25. plugin.stomp.user = mcollective
26. plugin.stomp.password = marionette
27.
28. # AgentPuppetd
29. plugin.puppetd.puppetd = /usr/sbin/puppetd
30. plugin.puppetd.lockfile = /var/lib/puppet/state/puppetdlock
31. plugin.puppetd.statefile = /var/lib/puppet/state/state.yaml
32. plugin.puppetd.pidfile = /var/run/puppet/agent.pid
33. plugin.puppetd.splaytime = 100
34. plugin.puppet.summary = /var/lib/puppet/state/last_run_summary.yaml
35.
36. #####
37. # FACTS #
38. #####
39.
40. factsource = facter
41. plugin.yaml = /etc/mcollective/facts.yaml
42. plugin.facter.facterlib = /var/lib/puppet/lib/facter
43. fact_cache_time = 300

```

7.22 Bind

Il peut être parfois utile d'avoir un serveur DNS de cache en locale pour accélérer certains traitement et ne pas être dépendant d'un DNS tierce s'il tombe pendant quelques instants. Pour plus d'infos sur Bind, suivez ce lien. Créons l'arborescence :



mkdir

```
0. mkdir -p /etc/puppet/modules/bind/{manifests,files,templates}
```

7.22.1 init.pp



/etc/puppet/modules/bind/manifests/init.pp

```

0. /*
1. Bind Module for Puppet
2. Made by Pierre Mavro
3. */
4. class bind {
5.   # Check OS and request the appropriate function
6.   case $::operatingsystem {
7.     'RedHat' : {
8.       include ::bind::redhat
9.     }
10.    #'sunos': { include packages_defaults::solaris }
11.    default : {
12.      notice("Module ${module_name} is not supported on ${::operatingsystem}")
13.    }
14.  }
15. }

```

7.22.2 redhat.pp



/etc/puppet/modules/bind/manifests/redhat.pp

```


0. /*
1. Bind Module for Puppet
2. Made by Pierre Mavro
3. */
4. class bind::redhat
5. {
6.   # Install bind package
7.   package {
8.     'bind' :
9.       ensure => present
10.  }
11.
12.   # resolv.conf file
13.   file {
14.     "/etc/resolv.conf" :
15.       source => "puppet:///modules/bind/resolvconf",
16.       mode => 744,
17.       owner => root,
18.       group => root,
19.       require => File['/etc/named.conf']
20.   }
21.
22.   # Bind main config file for cache server
23.   file {
24.     "/etc/named.conf" :
25.       content => template("bind/named.conf.$::operatingsystem"),
26.       mode => 640,
27.       owner => root,
28.       group => named,
29.       notify => Service["named"]
30.   }
31.
32.   # Service should start on boot and be running
33.   service {
34.     'named' :
35.       enable => true,
36.       ensure => running,
37.       require => [ Package['bind'], File['/etc/named.conf', '/etc/resolv.conf'] ]
38.   }

```

```
39. }
```

7.22.3 files

Nous allons gérer le resolv.conf ici :

 /etc/puppet/modules/bind/files/resolvconf

```
0. # Generated by Puppet
1. domain deimos.fr
2. search deimos.fr deimos.lan
3. nameserver 127.0.0.1
```

7.22.4 templates


Et pour finir, la configuration du template qui agira avec les informations renseignées dans vars.pp :



```
0. // Generated by Puppet
1. //
2. // named.conf
3. //
4. // Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
5. // server as a caching only nameserver (as a localhost DNS resolver only).
6. //
7. // See /usr/share/doc/bind*/sample/ for example named configuration files.
8. //
9.
10. options {
11.     listen-on port 53 { 127.0.0.1; };
12.     listen-on-v6 port 53 { ::1; };
13.     directory "/var/named";
14.     dump-file "/var/named/data/cache_dump.db";
15.     statistics-file "/var/named/data/named_stats.txt";
16.     memstatistics-file "/var/named/data/named_mem_stats.txt";
17.     allow-query { localhost; };
18.     recursion yes;
19.
20.     dnssec-enable no;
21.     dnssec-validation no;
22.     dnssec-lookaside auto;
23.
24.     forwarders {
25.         <% dns_servers.each do |dnsval| -%>
26.             <%= dnsval %>;
27.         <% end -%>
28.     };
29.
30.     /* Path to ISC DLV key */
31.     bindkeys-file "/etc/named.iscdlv.key";
32.
33.     managed-keys-directory "/var/named/dynamic";
34. };
35.
36. logging {
37.     channel default_debug {
38.         file "data/named.run";
39.         severity dynamic;
40.     };
41. };
42.
43. zone "." IN {
44.     type hint;
45.     file "named.ca";
46. };
47.
48. include "/etc/named.rfc1912.zones";
49. include "/etc/named.root.key";
```

7.23 Importation d'un module

Les modules doivent être importés dans puppet pour qu'ils soient pris en charge. Ici, nous n'avons pas à nous en soucier puisque d'après la configuration serveur que nous en avons faite, il va automatiquement charger les modules dans /etc/puppet/modules. Cependant si vous souhaitez autoriser module par module, il vous faut les importer à la main :

 /etc/puppet/manifests/modules.pp

```
0. # /etc/puppet/manifests/modules.pp
1.
2. import "sudo"
3. import "ssh"
```



WARNING

Faites attention car à partir du moment où ceci est renseigné, nul besoin de redémarrer puppet pour que les changements soient pris en compte. Il va donc falloir faire attention à ce que vous rendez disponible à l'instant t.

7.23.1 Importer tous les modules d'un coup

Ceci peut avoir de graves conséquences, mais sachez qu'il est possible de le faire :

```
etc/puppet/manifests/modules.pp
```

```
0. # /etc/puppet/manifests/modules.pp
1.
2. import "*"pp"
```

8 Utilisation

8.1 Certificats

Puppet travaille avec des certificats pour les échanges clients/serveurs. Il va donc falloir générer des clés SSL sur le serveur et faire un échange ensuite avec tous les clients. L'utilisation des clés SSL nécessite donc une bonne configuration de votre serveur DNS. Vérifiez donc bien à ce que :

- Le nom du serveur soit définitif
- Le nom du serveur soit accessible via puppet.mydomain.com (je continuerais la configuration pour moi avec puppet-prd.deimos.fr)

8.1.1 Création d'un certificat

- Nous allons donc créer notre certificat (**normalement déjà fait lorsque l'installation est faite sur Debian**) :

```
puppetca
```

```
puppetca -g puppet-prd.deimos.fr
```



WARNING

Il est impératifs que **TOUS les noms renseignés dans le certificat soient joignables**' sous peine que les clients ne puissent se synchroniser avec le serveur

- Dans le cas où vous souhaitez valider un certificat avec plusieurs noms de domaine, il va falloir procéder comme ceci :

```
puppetca
```

```
puppetca -g --dns_alt_names puppet:scar.deimos.fr puppet-prd.deimos.fr
```

Insérez les noms dont vous avez besoin les uns à la suite des autres. Je vous rappelle que par défaut, les clients vont chercher **puppet.mydomain.com**, donc n'hésitez pas à rajouter des noms si besoin.

Vous pouvez ensuite vérifier que le certificat contient bien les 2 noms :

```
openssl
```

```
> openssl x509 -text -in /var/lib/puppet/ssl/certs/puppet-prd.deimos.fr.pem | grep DNS
DNS:puppet, DNS:scar.deimos.fr, DNS:puppet-prd.deimos.fr
```

Vous pouvez voir d'éventuelles erreurs de certificats en vous connectant :

```
openssl
```

```
openssl s_client -host puppet -port 8140 -cert /path/to/ssl/certs/node.domain.com.pem -key /path/to/ssl/private_keys/node.domain.com.pem -CAfile /path/to/ssl/certs/ca.pem
```

Note: N'oubliez pas de redémarrer votre serveur web si vous faites le changement de certificats

8.1.2 Ajout d'un client puppet au serveur

Pour le certificat, c'est simple, nous allons faire une demande de certificats :

```
puppetd
```

```

$ puppetd --server puppet-prd.deimos.fr --waitforcert 60 --test
notice: Ignoring cache
info: Caching catalog at /var/lib/puppet/state/localconfig.yaml
notice: Starting catalog run
notice: //information_class/Exec[echo running on debian-puppet-prd.deimos.fr is a Debian with ip 192.168.0.106. Message is 'puppet c'est super']/returns: executed successfully
notice: Finished catalog run in 0.45 seconds
```

Maintenant nous allons nous connecter **sur le serveur** et lancer cette commande :

```
puppetca
```

```

$ puppetca -l
debian-puppet-prd.deimos.fr
```

Je vois ici par exemple que j'ai un host qui veut faire un échange de clefs afin ensuite d'obtenir les configurations qui lui sont dues. Seulement il va falloir l'autoriser. Pour ce faire :



```
puppetca -s debian-puppet-prd.deimos.fr
```

La machine est donc maintenant acceptée et peut aller chercher des confs sur le serveur Puppet.

Si on veut refuser un noeud qui est en attente :



```
puppetca -c debian-puppet-prd.deimos.fr
```

Si on veut voir la liste de tous les noeuds autorisés, puppetmaster les tient enregistrés ici :



```
/var/lib/puppet/ssl/ca/inventory.txt
```

```
0. 0x0001 2010-03-08T15:45:48GMT 2015-03-07T15:45:48GMT /CN=puppet-prd.deimos.fr
1. 0x0002 2010-03-08T16:36:16GMT 2015-03-07T16:36:16GMT /CN=node1.deimos.fr
2. 0x0003 2010-03-08T16:36:25GMT 2015-03-07T16:36:25GMT /CN=node2.deimos.fr
3. 0x0004 2010-04-14T12:41:24GMT 2015-04-13T12:41:24GMT /CN=node3.deimos.fr
```

8.1.3 Synchroniser un client

Maintenant que nos machines sont connectées, nous allons vouloir les synchroniser de temps à autre. Voici donc comment faire une synchronisation manuelle depuis une machine cliente :



```
puppet
```

```
0. puppet agent -t
```

Si vous souhaitez synchroniser uniquement un module, vous pouvez utiliser l'option --tags :



```
puppet
```

```
0. puppet agent -t --tags module1 module2...
```

8.1.3.1 Simuler

Si vous avez besoin de tester avant de déployer réellement, il existe l'option --noop :



```
puppet
```

```
0. puppet agent -t --noop
```

Vous pouvez également ajouter dans un manifest la directive 'audit' si vous souhaitez simplement auditer une ressource :

```
0. file { ["/etc/passwd"] :
1.   audit => [ owner, mode ],
2. }
```

Ici on demande donc d'auditer l'utilisateur et les droits, mais sachez qu'il est possible de remplacer par 'all' pour tout auditer !

8.1.4 Révoquer un certificat

- Si vous souhaitez révoquer un certificat, voici comment procéder sur le serveur :



```
puppet
```

```
0. puppet cert clean ma_machine
```

- Sinon il existe cette méthode :



```
puppetca
```



```
puppetca -r ma_machine
puppetca -c ma_machine
```

Simple non ? :-)

Si vous souhaitez réassigner de nouveau, supprimez côté client le dossier ssl dans `/etc/puppet/` ou `/var/lib/puppet/`. Ensuite vous pouvez relancer une demande de certificat.

8.1.5 Révoquer tous les certificats du serveur

Si vous avez votre puppet master qui est cassé de partout et voulez régénérer de nouvelles clefs et supprimer toutes les anciennes :

```
0. puppetca clean --all
1. rm -Rf /var/lib/puppet/ssl/certs/*
2. /etc/init.d/puppetmaster restart
```

8.2 Surveillance des processus

Sur les clients, il n'est pas nécessaire d'utiliser un logiciel de surveillance, puisque les daemons puppetd ne sont lancés qu'à la main ou par tâche CRON. Sur le serveur, il est important que le processus puppetmaster soit toujours présent. On pourra utiliser le logiciel 'monit', qui pourra redémarrer automatiquement le processus en cas de problème.

8.2.1 Détermination de l'état des noeuds

Pour voir s'il y a des problèmes sur un noeud, on pourra lancer manuellement la commande suivante :

```
puppetd
```

```
puppetd --no-daemon --verbose --onetime
```

Si l'on souhaite connaître le dernier état/résultat d'une mise à jour puppet, on pourra utiliser le système de 'report' une fois activé (sur les clients) :

```
/etc/puppet/puppet.conf
```

```
...
report = true
...
```

En activant le reporting, à chaque exécution du daemon puppetd, un compte rendu sera envoyé sur le puppetmaster dans un fichier au format YAML, dans le répertoire `/var/lib/puppet/reports/NOM_MACHINE`.

Voici un exemple de fichier de report, facilement transformable en dictionnaire avec le module yaml de python (ou ruby) :

```
/var/lib/puppet/reports/deb-puppet-client.deimos.fr
```

```
0. --- !ruby/object:Puppet::Transaction::Report
1. host: deb-puppet-client.deimos.fr
2. logs:
3. - !ruby/object:Puppet::Util::Log
4.   level: :info
5.   message: Applying configuration version '1275982371'
6.   source: Puppet
7.   tags:
8.   - info
9.   time: 2010-06-08 11:37:59.521132 +02:00
10. - !ruby/object:Puppet::Util::Log
11.   file: /etc/puppet/modules/sudo/manifests/init.pp
12.   level: :error
13.   line: 11
14.   message: "Failed to retrieve current state of resource: Could not retrieve information from source(s) puppet:///sudo/sudoers at /etc/puppet/modules/sudo/manifests/init.pp:11"
15.   source: //sudo/File[/etc/sudoers]
```

Une méthode plus jolie par interface graphique existe également, il s'agit du Puppet Dashboard.

9 Utilisation avancée

9.1 Vérifier la syntaxe de ses .pp

Lorsque vous créez/éditez un module puppet, il peut être vite très pratique de vérifier la syntaxe. Voici comment faire :

```
puppet
```

```
0. puppet parser validate init.pp
```

Et si vous souhaitez le faire à plus grande échelle :



find

```
0. find /etc/puppet/ -name '*.pp' | xargs -n 1 -t puppet parser validate
```

9.2 Outrepasser des restrictions

Si par exemple, nous avons défini une classe et que pour certains hôtes, nous souhaitons modifier cette configuration, nous devons faire comme ceci :



```
0. class somehost_postfix inherits postfix {
1.   # blah blah blah
2. }
3.
4. node somehost {
5.   include somehost_postfix
6. }
```

Admettons que nous avons un module postfix de défini. Nous souhaitons appliquer une config particulière à certains hosts défini ici par 'somehost'. Pour bypasser la configuration, il faut créer une classe somehost_postfix'. J'insiste ici sur la nomenclature du nom à donner pour cette classe puisque c'est uniquement comme cela que Puppet reconnaitra que vous souhaitez appliquer une configuration particulière.

9.3 Désactiver une ressource

Pour désactiver temporairement une ressource, il suffit de mettre noop à true :

```
0. file { ["/etc/passwd"]:
1.   noop => true
2. }
```

9.4 Pre et Post puppet run

Il est possible de lancer des scripts avant et après l'exécution d'un Puppet run. Cela peut s'avérer utile dans le cas d'une sauvegarde de certains fichiers via etckeeper par exemple. Ajoutez ceci dans le fichier puppet.conf de vos clients :



/etc/puppet/puppet.conf

```
0. [...]
1. prerun_command = /usr/local/bin/before-puppet-run.sh
2. postrun_command = /usr/local/bin/after-puppet-run.sh
```

9.5 CFT

CFT (<http://cft.et.redhat.com/>) (prononcez shift) est un petit logiciel qui va regarder ce que vous faites pendant une période donnée pour vous générer un manifest. Par exemple, vous le lancez juste avant de faire une installation avec sa configuration et il va vous générer le manifest une fois terminé. Un exemple vaut mieux qu'un long discours :



cft

```
0. cft begin apache
1. [...]
2. cft finish apache
3. cft manifest apache
```

9.6 Générer un manifest depuis un système existant

Voici une solution simple de générer depuis un système déjà installé un manifest en lui spécifiant une ressource. Voici quelques exemples :



puppet

```
0. puppet resource user root
1. puppet resource service httpd
2. puppet resource package postfix
```

9.7 Puppet Push

Puppet fonctionne en mode client -> serveur. C'est le client qui contacte toutes les 30 min (par défaut) le serveur et demande une synchronisation. Lorsque vous êtes en mode boulet ou bien quand vous souhaitez déclencher à un instant t la mise à jour de vos machines clientes vers le serveur, il y a un mode particulier (listen). Le problème c'est que le client puppet ne peut pas tourner en mode client et listen. Il faut donc 2 instances... bref les cauchemars commencent.

Pour palier à ce problème, j'ai donc développé Puppet push qui permet de demander aux clients (via SSH) de se synchroniser. Vous l'aurez compris, il est plus que nécessaire d'avoir un échange de clef effectué au préalable avec les clients. Comment faire ? Pas de soucis, nous avons vu cela plus haut.

Pour accéder à la dernière version, suivez ce lien : http://www.deimos.fr/gitweb/?p=puppet_push.git;a=tree

9.8 MCollective

MCollective c'est un peu l'usine à gaz, mais c'est très puissant et fonctionne très bien avec Puppet. Il vous permet comme Puppet Push de faire plusieurs actions sur des noeuds, mais utilise un protocole dédié, il n'a pas besoin d'SSH pour communiquer avec les noeuds. Pour en savoir plus, regardez cet article.

10 FAQ

10.1 err: Could not retrieve catalog from remote server: hostname was not match with the server certificate

Vous avez un problème avec vos certificats. Le mieux c'est de régénérer un certificats avec tous les hostname du serveur dans ce certificat : Création d'un certificat

11 Ressources

<http://reductivelabs.com/products/puppet/>

<http://www.rottenbytes.info>

Modules pour Puppet (<http://git.black.co.at/>)

Puppet recipes (<http://reductivelabs.com/trac/puppet/wiki/Recipes>)

Types d'objets pour Puppet (<http://reductivelabs.com/trac/puppet/wiki/TypeReference>)

Puppet SSL Explained (<http://www.masterzen.fr/2010/11/14/puppet-ssl-explained/>) (PDF)

<http://puppetcookbook.com/>

Récupérée de « http://wiki.deimos.fr/index.php?title=Puppet:_Solution_de_gestion_de_fichier_de_configuration&oldid=11874 »